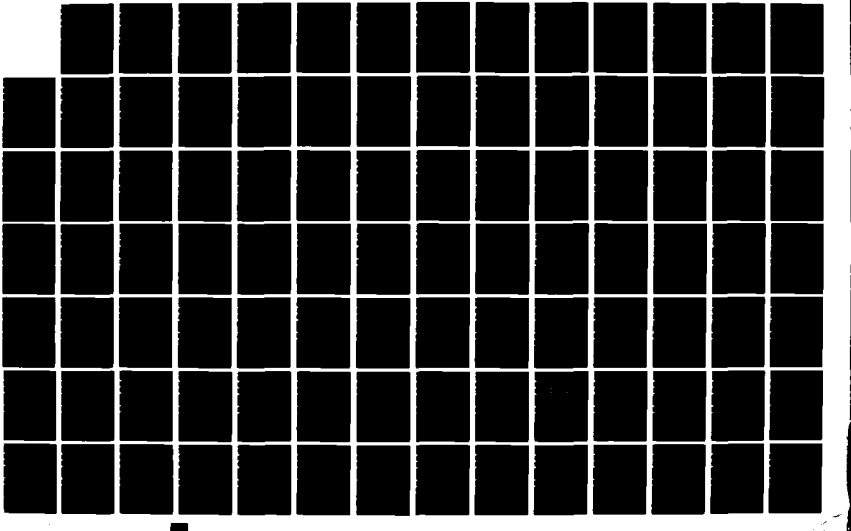
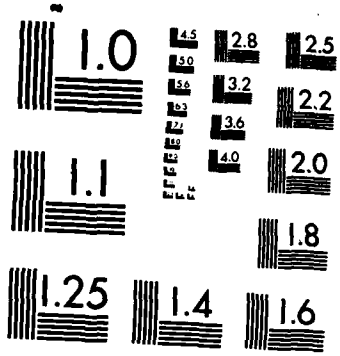
AD-A164 111 LQG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING G L GROSS DEC 85 1/2

UNCLASSIFIED AFIT/GAE/ENG/85D-1 F/G 17/7 NL





LQG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15

THESIS

Gregory L. Gross
Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GAE/ENG/85D-1

LQG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15

THESIS

Gregory L. Gross
Captain, USAF

AFIT/GAE/ENG/85D-1

Approved for public release; distribution unlimited

AFIT/GAE/ENG/85D-1

LQG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Aeronautical Engineering

Gregory L. Gross, B.S., M.B.A.
Capt USAF

December 1985

ACKNOWLEGEMENTS

I would like to express my sincere appreciation to my thesis advisor, Dr. Peter S. Maybeck, for his enthusiasm and guidance. He made this project an incredible learning experience and definitely is responsible for the quality of the final report.

I would also like to thank the GE-85D guidance and control students for their assistance in my research. Special thanks go to Lt. Bob Houston for keeping my efforts headed in the right direction and providing a good sounding board for my ideas.

Last, and by no means least, a tribute to my wife, Elizabeth, who bore the role of AFIT widow and section leader's wife so very well. Her help in preparation of the final copy was invaluable.

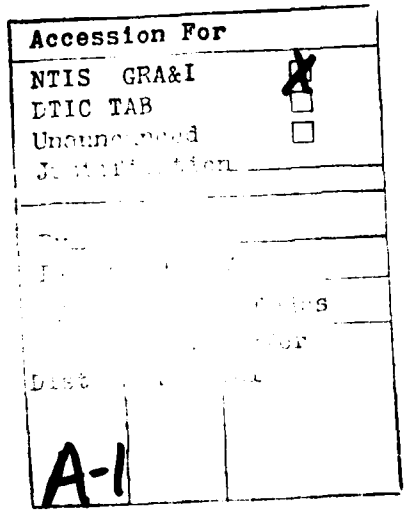


Table of Contents

LIST OF FIGURES

Abstract

A robust controller for the approach and landing phase of the Short Take-off and Landing (STOL) F-15 is developed via LQG/LTR (Linear System model, Quadratic cost, Gaussian models of uncertainty, used for controller synthesis, with Loop Transmission Recovery techniques of tuning the filter in the loop for control robustness enhancement) methods. Reduced-order full-state feedback controllers are synthesized using CGT/PI (Command Generator Tracking feedforward compensator to incorporate handling qualities, with Proportional plus Integral feedback) synthesis, specifically using implicit model following to provide good robustness characteristics in the full-state feedback case. The robustness is fully assessed using realistic simulations of the real-world system with meaningful deviations from design conditions. Once a Kalman filter is embedded into the loop to estimate states rather than assuming artificial access to all states, LTR methodology is used to preserve as much robustness as possible. A full assessment of performance and robustness of these final implementable designs is provided.

LQG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15

I. INTRODUCTION

1-1 Background

Aircraft flight control laws must be designed so that the aircraft will achieve satisfactory response not only to design conditions, but also to real-world situations. The differences between the real-world conditions and the mathematical model used in the development of the controller stem from the fact that there are not really n^{th}-order systems in nature, only n^{th}-order models of physical phenomena. The controller is usually designed from a model of reduced order from the original system model and, in general, ignores nonlinearities. Once designed, the controller will face unmodelled disturbances on the system and variations in the parameters of the system from the design conditions. A controller that, in the face of the above conditions, yields a closed-loop system that is stable and retains some measure of performance, is termed a robust controller. This is one of the main goals in flight control design.

Currently there are several approaches that are in use in control design. Many use time-domain techniques, which call for a system modelled as linear and manipulated

under linear system theory to develop control laws to be implemented on a digital computer [1,3,8,27]. Other approaches invoke the frequency- domain theory of classical control techniques and the benefit of readily understood graphical stability analysis [7,8,13]. The design method employed in this thesis combines the ease of implementation of the time domain techniques with the availability of stability and stability robustness insights of frequency domain techniques.

This method assumes that the system to be controlled can be modelled as a linear system that may have dynamics disturbances or measurement corruptions represented as Gaussian processes, with a scalar quadratic cost function to be minimized for controller synthesis and is known as Linear-Quadratic-Gaussian (LQG) design [20]. The controller to be designed will be a Command Generator Tracker (CGT) which provides feedforward control, a Proportional plus Integral (PI) controller to provide feedback control, and a Kalman Filter to estimate the states in the face of incomplete/noise-corrupted measurements. The PI controller will be developed using implicit model following control techniques, so that it will be as robust as possible [24]. The Kalman filter, which has a negative effect on the robustness of the controller, is then tuned using Loop Transfer Recovery procedures [9,10,15,28] to recover as much of the robustness

of the corresponding full-state feedback controller as possible.

1-2 Problem

The purpose of this study is to develop a robust flight controller for the approach and landing phase for the Short Takeoff and Landing (STOL) F-15. This digital controller is designed using LQG design techniques with the CGT/PI/KF control laws developed with implicit model following to enhance robustness and LTR techniques used to attempt to recover robustness lost to the Kalman filter. It is required that the controller, and thus the aircraft, be stable and maintain a minimum level of performance in the face of large parameter variations, such as those occurring from off-design operation; actuator failure; actuator saturations; or mismodelled actuator dynamics, either from errors in the model or from the reduction in order for the simplification of the mathematical model.

This study has been done concurrently with efforts using other control design methods for the same flight arena [1,7] and with studies using this method and one other for up and away flight manuevers [14,27]. While it is inevitable that comparisons between the merits of the various methods will be made, the primary goal of this research is not to prove which method is superior, but to exploit and enhance controller design methodology.

1-3 Scope

This thesis effort will design and analyze a CGT/PI/KF controller for the approach and landing phase of flight STOL F-15 using LQG/LTR synthesis techniques via a software package originally developed by Capt R. Floyd [11]. The controller has been designed about a nominal condition of 119 knots at sea level with a light gross weight and in a landing configuration. This controller is subjected to robustness analysis and, if required, robustness enhancement. The analysis is accomplished by means of a linear covariance analysis tool developed by Lt. A. Moseley [25] and by a nonlinear analysis tool originally designed by Capt. W. Miller [24] and modified for this study. Robustness enhancement will be attempted using the discrete-time modifications of techniques developed for the robustness enhancement of continuous-time regulators by Doyle and Stein [9,10].

1-4 Sequence of Presentation

Chapter 2 of this thesis develops the CGT/PI controller, starting with a proportional regulator, then adding the integral characteristics to get the PI controller. These are then combined with the CGT to form a CGT/R and CGT/PI controllers, followed by a discussion of model following controllers. Chapter 3 adds the Kalman filter to the CGT/PI/KF controller and discusses the impact of the filter on robustness as well as a robustness

enhancement technique. Chapter 4 presents the system models used in the controller design for the STOL F-15. Chapter 5 demonstrates the controller design process and performance analysis used in this study. Controller designs and the analysis of these designs are presented in this chapter. The last chapter, Chapter 6, contains the conclusions drawn from this study and gives recommendations for further study.

II. LQG CONTROLLER THEORY

2-1 INTRODUCTION

This chapter shows the development of optimal Linear Quadratic Gaussian (LQG) controllers for continuous-time systems having sampled-data measurements. The development starts with a full-state feedback model, as will be described in Chapter 4 for this particular application, assuming perfect measurements. The inaccessibility of all states and imperfection of measurements will be considered in the next chapter.

The three controllers to be developed are the proportional regulator, the proportional plus integral controller (PI) and the open loop Command Generator Tracker (CGT). Either the proportional regulator or the PI controller can be combined with the CGT and a Kalman Filter to provide the controller for the aircraft.

2-2 PROPORTIONAL REGULATOR

A continuous-time system is assumed to be described by means of the linear stochastic differential equation

$$\dot{\underline{x}}(t) = F(t)\,\underline{x}(t) + B(t)\,\underline{u}(t) + G(t)\,\underline{w}(t) \qquad (2\text{-}1)$$

where $\underline{w}(t)$ is a zero mean white Gaussian noise with

$$E\{\underline{w}(t)\,\underline{w}^T(t+\tau)\} = Q(t)\delta(\tau) \qquad (2\text{-}2)$$

The aircraft considered in this thesis is a continuous-time system with sample-data measurements. It can be modelled as an equivalent discrete-time model [19], i.e., the solution to the differential equation (2-1), by

the linear difference equation:

$$\underline{x}(t_{i+1}) = \Phi(t_{i+1},t_i)\underline{x}(t_i) + B_d(t_i)\underline{u}(t_i) + G_d(t_i)\underline{w}_d(t_i) \quad (2\text{-}3)$$

where $\underline{w}_d(t_i)$ is a zero-mean discrete-time white Gaussian noise with

$$E[\underline{w}_d(t_i)\underline{w}_d^T(t_j)] = Q_d(t_i)\,\delta_{ij} \quad (2\text{-}4)$$

The Φ matrix is the state transition matrix associated with F and B_d, G_d, and Q_d are defined in terms of Φ, B, G, and Q, as is shown in Reference 19.

An optimal discrete-time LQG controller minimizes the cost functional

$$J = E\left\{ \frac{1}{2}\underline{x}^T(t_{n+1})\,X_f\,\underline{x}(t_{n+1}) + \sum_{i=0}^{n} \frac{1}{2}\begin{bmatrix}\underline{x}(t_i)\\ \underline{u}(t_i)\end{bmatrix}^T\begin{bmatrix}X(t_i) & S(t_i)\\ S(t_i)^T & U(t_i)\end{bmatrix}\begin{bmatrix}\underline{x}(t_i)\\ \underline{u}(t_i)\end{bmatrix} \right\} \quad (2\text{-}5)$$

where t_n is the last time at which a control may be applied, and a zero-order hold is used to keep the control inputs constant over each time interval. The weighting matrices X_f and $X(t_i)$ are positive semi-definite and the $U(t_i)$ weighting matrix is positive definite. The X_f matrix is used to control the cost at the final time due to state deviations from desired values (zero for perturbation states) and the X matrix weights the state deviations at the individual sample times. The U matrix is the weighting matrix that determines the cost of applying controls to the states. The $S(t_i)$ matrix is chosen so that the symmetric composite matrix in Equation (2-3) is positive semi-

definite and is used to weight the cost of cross-terms. This cross-term matrix's main use is to exert control over the entire interval and not just at the sample times as would occur with $S(t_i) = 0$ [20].

For this problem, the LQ optimal full-state feedback control law is given by

$$\underline{u}^*(t_i) = -G_c^*(t_i)\ \underline{x}(t_i) \qquad (2\text{-}6)$$

where $G_c^*(t_i)$ is the solution to

$$G_c^*(t_i) = [U+B_d^T\ K_c(t_{i+1})B_d]^{-1}[B_d^T K_c(t_{i+1})\Phi+S^T] \qquad (2\text{-}7)$$

and

$$K_c(t_i) = X + \Phi^T K_c(t_{i+1})\Phi - [B_d^T K_c(t_{i+1})\Phi+S^T]^T G_c^*(t_i) \qquad (2\text{-}8)$$

which is a backward running Riccati difference equation with a final condition

$$K_c(t_{n+1}) = X_f \qquad (2\text{-}9)$$

This controller is represented by the block diagram in Figure 2-1.

2-3 PROPORTIONAL PLUS INTEGRAL (PI) CONTROLLERS

The preceding regulator may not be satisfactory when a nonzero steady-state control must be applied with a zero input to the controller. For example, when the input to the controller is the tracking error between desired and actual values of the controlled variables, even when this input is zero, we want the controller to generate the (generally nonzero) output to maintain the system at equilibrium conditions to maintain zero tracking error. This can be achieved if the controller includes integral

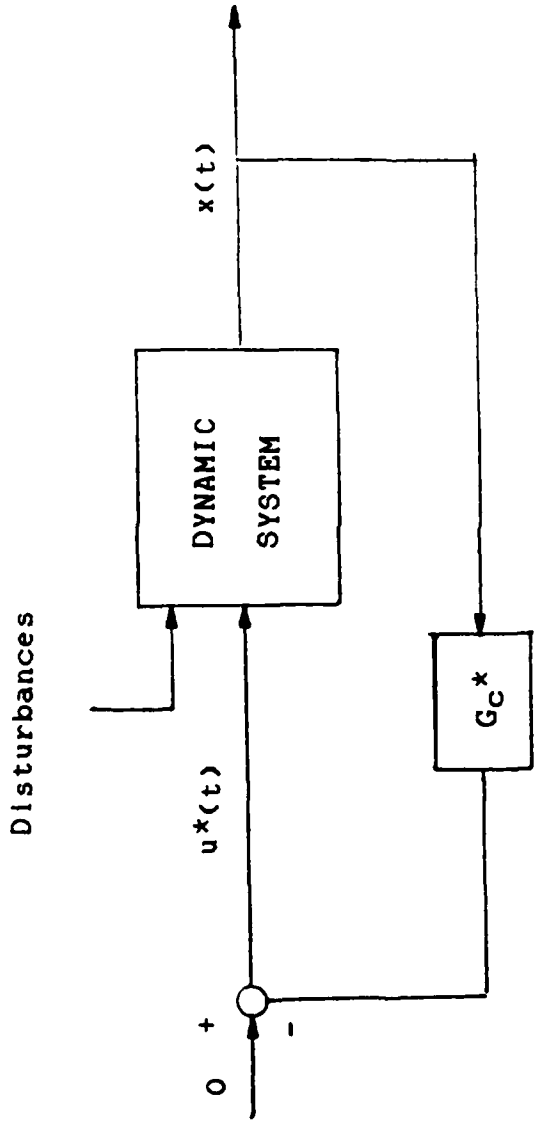


Figure 2-1. Proportional Regulator

action, which is especially beneficial in generating zero steady state mean tracking errors in the face of unmodelled constant disturbances. This desire to achieve the "type-1 property" [20] motivates the use of a PI controller (Fig. 2-2) with its summation (or pseudo-integration) of the regulation error ($\underline{y}_d(t_i) - \underline{y}_c(t_i)$) in controlled variables, where $\underline{y}_c$ is the controlled variable and $\underline{y}_d$ is the reference signal which provides the input to the controller.

The incremental form of the PI controller will be implemented in this thesis. In this form of the controller, only the changes in the states and commands, and not the states and commands themselves, are used to generate increments in control relative to the value at the previous sample instant. This form is preferable, as initial conditions for the controller states are not required and relinearization about nominal values is easier in applications where the system is nonlinear. This form also lends itself more readily to anti-windup compensation [20], which is an important consideration for flight controls that are easily saturated.

2-3.1 Control-Rate Pseudo-Integration for Sampled-Data Systems

In Section 2-2, the equivalent discrete-time difference equation was derived from a continuous-time system model and took the form

$$\underline{x}(t_{i+1}) = \Phi\underline{x}(t_i) + B_d\underline{u}(t_i) \qquad (2\text{-}10)$$

with the noise vector deleted by virtue of the certainty

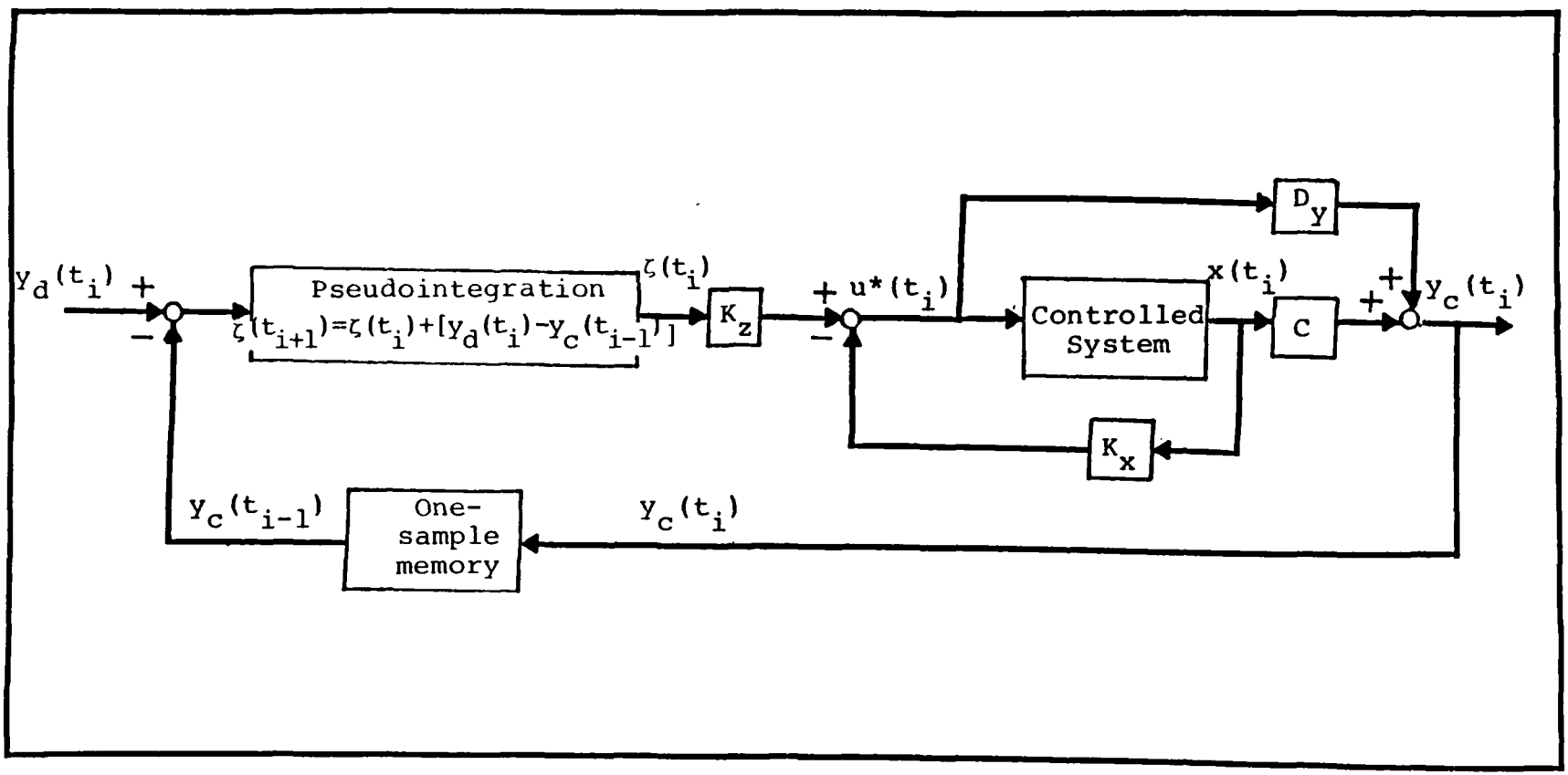


Figure 2-2. PI Controller

equivalence principle. This principle states that, in the generation of an LQG controller, the controller gains can be evaluated on the basis of the corresponding deterministic optimal LQ control problem, and then the full-state feedbacks are replaced with Kalman filter state estimates [20].

For this thesis, we will use perturbation state and control variables, $\delta\underline{x}$ and $\delta\underline{u}$, respectively, defined as

$$\delta\underline{x}(t_1) = \underline{x}(t_1) - \underline{x}_o \qquad (2\text{-}11)$$

$$\delta\underline{u}(t_1) = \underline{u}(t_1) - \underline{u}_o \qquad (2\text{-}12)$$

where $\underline{x}_o$ and $\underline{u}_o$ are the nominal values of the states and controls that maintain the system at the equilibrium trim operating condition to maintain desired output values. Assuming that the system output is a linear combination of the states and controls, we can write:

$$\underline{y}_c(t_1) = C\underline{x}(t_1) + D_y\underline{u}(t_1) \qquad (2\text{-}13)$$

To find the nominal control, $\underline{u}_o$, that will hold the system at the equilibrium operating point, such that $\underline{y}_c$ equals the desired system output, $\underline{y}_d$, Equations (2-10) and (2-13) yield

$$\underline{x}_o = \Phi\underline{x}_o + B_d\underline{u}_o \qquad (2\text{-}14a)$$

$$\underline{y}_d = C\underline{x}_o + D_y\underline{u}_o \qquad (2\text{-}14b)$$

or

$$\begin{bmatrix} \underline{0} \\ \underline{y}_d \end{bmatrix} = \begin{bmatrix} (\Phi - I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} \underline{x}_o \\ \underline{u}_o \end{bmatrix} \qquad (2\text{-}15)$$

which must be solved for $\underline{x}_o$ and $\underline{u}_o$ in terms of $\underline{y}_d$. The solution to Equation (2-15) yields

$$\begin{bmatrix} \underline{x}_o \\ \underline{u}_o \end{bmatrix} = \begin{bmatrix} (\Phi - I) & B_d \\ C & D_y \end{bmatrix}^{-1} \begin{bmatrix} \underline{0} \\ \underline{Y}_d \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} \underline{0} \\ \underline{Y}_d \end{bmatrix} \quad (2\text{-}16)$$

Therefore,

$$\underline{x}_o = W_{12}\underline{Y}_d \qquad \underline{u}_o = W_{22}\underline{Y}_d \quad (2\text{-}17)$$

We can write Equation (2-12) for time t_i and subtract it from Equation (2-12) for time t_{i+1} to get

$$\delta\underline{u}(t_{i+1}) = \delta\underline{u}(t_i) + [\underline{u}(t_{i+1}) - \underline{u}(t_i)] \quad (2\text{-}18)$$

which is the discrete-time equivalent of an integration. Defining

$$\Delta\underline{u}(t_i) = \underline{u}(t_{i+1}) - \underline{u}(t_i) \quad (2\text{-}19)$$

as the control pseudo-rate, Equation (2-18) becomes

$$\delta\underline{u}(t_{i+1}) = \delta\underline{u}(t_i) + \Delta\underline{u}(t_i) \quad (2\text{-}20)$$

Using Equations (2-10) and (2-20), an augmented state equation can be written for the perturbation variables $\delta\underline{x}(t_i)$ and $\delta\underline{u}(t_i)$ with the control pseudo-rate as the input:

$$\begin{bmatrix} \delta\underline{x}(t_{i+1}) \\ \delta\underline{u}(t_{i+1}) \end{bmatrix} = \begin{bmatrix} \Phi & B_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \delta\underline{x}(t_i) \\ \delta\underline{u}(t_i) \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} \Delta\underline{u}(t_i) \quad (2\text{-}21)$$

2-3.2 Achieving Type-1 Control

In her thesis on robust flight controllers [15], Lt Jean Howey employs previously developed theory [20] to show that the desired integral characteristics can be achieved by manipulating the discrete-time equivalent of

$$\underline{u}^*(t) = -K_x\,\underline{x}(t) + K_z \int_{to}^{t} [\underline{y}_d - \underline{y}_c(\tau)]\, d\tau \quad (2\text{-}22)$$

which is given by

$$\underline{u}^*(t_i) = -K_x\,\underline{x}(t_i) + K_z \sum_{j=-1}^{i-1} [\underline{y}_d - \underline{y}_c(t_j)] \qquad (2\text{-}23)$$

This discrete-time equivalent equation is then put into incremental form and rewritten as a perturbation equation to yield:

$$\begin{aligned}\delta\underline{u}^*(t_{i+1}) = \delta\underline{u}^*(t_i) &- K_x[\delta\underline{x}(t_{i+1}) - \delta\underline{x}(t_i)] \\ &- K_z\,[C\delta\underline{x}(t_i) + D_y\delta\underline{u}(t_i)]\end{aligned} \qquad (2\text{-}24)$$

The above result is then compared to the optimal regulator solution shown in Section 2-2 as augmented by the control pseudo-rates and subject to a quadratic cost criterion:

$$\begin{aligned}J_c = &\frac{1}{2}\begin{bmatrix}\delta\underline{x}(t_{n+1})\\ \delta\underline{u}(t_{n+1})\end{bmatrix}^T\begin{bmatrix}X_f & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\delta\underline{x}(t_{n+1})\\ \delta\underline{u}(t_{n+1})\end{bmatrix} \\ &+\sum_{i=-1}^{n}\frac{1}{2}\begin{bmatrix}\delta\underline{x}(t_i)\\ \delta\underline{u}(t_i)\\ \Delta\underline{u}(t_i)\end{bmatrix}^T\begin{bmatrix}X_{11} & X_{12} & S_1\\ X_{12}^T & X_{22} & S_2\\ S_1^T & S_2^T & U\end{bmatrix}\begin{bmatrix}\delta\underline{x}(t_i)\\ \delta\underline{u}(t_i)\\ \Delta\underline{u}(t_i)\end{bmatrix}\end{aligned} \qquad (2\text{-}25)$$

where X_{11} is the weighting matrix associated with state deviations from the nominal state trajectory, X_{22} weights control deviations from $\underline{u}_o$, and U is the weighting matrix associated with control pseudo-rates. This U matrix gives the designer a tool with which to place a weighting on the rate of change of control inputs and thereby enable him to iterate on the design readily so as to prevent the system from commanding actuator rates greater than physically realizable. The X_f matrix weights deviations of the state at the final time. The cross terms, X_{12}, S_1, and S_2, are a

by-product of the conversion of the system from continuous-time cost to discrete-time cost. This development is shown in References 10 and 14. The index for the summation in the immediately preceding cost equation begins at -1 rather than zero to provide an acceptable transient response to a change in setpoint, which may be large compared to succeeding control differences [15].

Applying the optimal regulator solution to Equation (2-25) and achieving a constant-gain control law by allowing $n \rightarrow \infty$, the steady-state control law is generated as

$$\Delta \underline{u}^*(t_i) = -G_{c1}^* \, \underline{\delta x}(t_i) - G_{c2}^* \, \underline{\delta u}(t_i) \qquad (2\text{-}26)$$

which, when combined with Equation (2-20), yields:

$$\underline{\delta u}^*(t_{i+1}) = \underline{\delta u}^*(t_i) - G_{c1}^* \, \underline{\delta x}(t_i) - G_{c2}^* \, \underline{\delta u}^*(t_i) \qquad (2\text{-}27)$$

$$= \underline{\delta u}^*(t_i) - [G_{c1}^* \; G_{c2}^*] \begin{bmatrix} \underline{\delta x}(t_i) \\ \underline{\delta u}(t_i) \end{bmatrix} \qquad (2\text{-}28)$$

Rewriting Equation (2-24) with the upper partition of Equation (2-21) yields:

$$\underline{\delta u}(t_{i+1}) = \underline{\delta u}(t_i) - [K_x \; K_z] \begin{bmatrix} (\Phi - I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} \underline{\delta x}(t_i) \\ \underline{\delta u}(t_i) \end{bmatrix} \qquad (2\text{-}29)$$

which, when compared with Equation (2-28), shows that

$$[K_x \; K_z] = [G_{c1}^* \; G_{c2}^*] \begin{bmatrix} (\Phi - I) & B_d \\ C & D_y \end{bmatrix}^{-1} \qquad (2\text{-}30)$$

and therefore

$$[K_x \; K_z] = [G_{c1}^* \; G_{c2}^*] \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \qquad (2\text{-}31)$$

This equation gives feedback gains as follows:

$$K_x = G_{c1}^* \, w_{11} + G_{c2}^* \, w_{21} \qquad (2\text{-}32a)$$

$$K_z = G_{c1}^* w_{12} + G_{c2}^* w_{22} \qquad (2\text{-}32b)$$

where G_{c1}^* and G_{c2}^* are found via the augmented state regulator solution and w_{11}, w_{12}, w_{21}, and w_{22} are defined in Equation (2-16). This leads to the incremental form of the PI controller:

$$\underline{\delta u}^*(t_{i+1}) = \underline{\delta u}^*(t_i) - K_x [\underline{\delta x}(t_{i+1}) - \underline{\delta x}(t_i)] + K_z [\underline{y}_d(t_{i+1}) - \underline{y}_c(t_i)] \qquad (2\text{-}33)$$

which achieves the desired Type-1 control. It is proper that the time indices for the last two terms do not match [20].

2-4 COMMAND GENERATOR TRACKER

This section presents the fundamentals of Command Generator Tracking, a formalization of numerous model following concepts. For a more complete development of these controllers, see also Reference 20 and the work of J. R. Broussard [5,6].

For a given system to maintain desired trajectories in real time, it must respond appropriately to commanded inputs and must reject disturbances. Both the desired trajectories and the disturbances to be rejected are formulated as the outputs of linear system models. The command generator model is defined as

$$\dot{\underline{x}}_m(t) = A_m \underline{x}_m(t) + B_m \underline{u}_m(t) \qquad (2\text{-}34)$$

with the output equation of

$$\underline{y}_m(t) = C_m \underline{x}_m(t) + D_m \underline{u}_m(t) \qquad (2\text{-}35)$$

In the discrete form, these equations become

$$\underline{x}_m(t_{i+1}) = \Phi_m \underline{x}_m(t_i) + B_{dm}\underline{u}_m(t_i) \quad (2\text{-}36)$$

and

$$\underline{y}_m(t_i) = C_m \underline{x}_m(t_i) + D_m \underline{u}_m(t_i) \quad (2\text{-}37)$$

Note that the C_m and D_m in Equation (2-37) can be different from those in Equation (2-35) to reflect the desire to control variables over the entire sample period and not just at the sample times [20].

The objective of the CGT controller is to force the output of the system model to match the output of the command model,

$$\underline{y}_c(t_i) = \underline{y}_m(t_i) \quad (2\text{-}38)$$

where $\underline{y}_c(t_i)$ is the output of the linear time-invariant system

$$\underline{x}(t_{i+1}) = \Phi \underline{x}(t_i) + B_d \underline{u}(t_i) + E_x \underline{n}_d(t_i) \quad (2\text{-}39a)$$

$$\underline{y}_c(t_i) = C \underline{x}(t_i) + D_y \underline{u}(t_i) + E_y \underline{n}_d(t_i) \quad (2\text{-}39b)$$

In the above equations, $\underline{n}_d$ is a disturbance modeled by

$$\underline{n}_d(t_{i+1}) = \Phi_n \underline{n}_d(t_i) + B_{dn}\underline{n}_{cmd}(t_i) + G_{dn}\underline{w}_{dn}(t_i) \quad (2\text{-}40)$$

where $\underline{n}_{cmd}$ is the commanded input to the disturbance shaping filter (here assumed to be zero for simplicity), and $\underline{w}_{dn}(.,.)$ is a white Gaussian noise with mean of zero and a covariance Q_{dn}.

The error that arises from deviations at any time t_i is

$$\underline{e}(t_i) = \underline{y}_c t_i) - \underline{y}_m(t_i) \quad (2\text{-}41a)$$

or, from Equations (2-13) and (2-37),

$$\underline{e}(t_i) = [C \; D_y \; E_y] \begin{bmatrix} \underline{x}(t_i) \\ \underline{u}(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} - [C_m \; D_m] \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \end{bmatrix} \quad (2\text{-}41b)$$

When this error is zero, the system is said to be tracking the "ideal state trajectory."[11] This ideal system must be defined so as to satisfy the original state and output equations and so must take the form

$$\underline{x}_I(t_{i+1}) = \Phi\underline{x}_I(t_i) + B_d\underline{u}_I(t_i) + E_x\underline{n}_d(t_i) \qquad (2\text{-}42a)$$

$$\underline{y}_I(t_i) = C\underline{x}_I(t_i) + D_y\underline{u}_I(t_i) + E_y\underline{n}_d(t_i) \qquad (2\text{-}42b)$$

where $\underline{x}_I$ and $\underline{y}_I$ are the ideal state and output vectors, respectively. The ideal system must also maintain zero error between the system and command model outputs, (ie. $\underline{e}(t_i) = \underline{0}$), so that $\underline{y}_I$ equals $\underline{y}_m$ for all time t_i, or

$$[C\ D_y\ E_y]\begin{bmatrix}\underline{x}_I(t_i)\\ \underline{u}_I(t_i)\\ \underline{n}_d(t_i)\end{bmatrix} = [C_m\ D_m]\begin{bmatrix}\underline{x}_m(t_i)\\ \underline{u}_m(t_i)\end{bmatrix} \qquad (2\text{-}43)$$

The ideal plant response must also, by assumption, be a linear function of the model state, model control, disturbance state, and, if applicable, disturbance control input: [15]

$$\begin{bmatrix}\underline{x}_I(t_i)\\ \underline{u}_I(t_i)\end{bmatrix} = \begin{bmatrix}A_{11} & A_{12} & A_{13}\\ A_{21} & A_{22} & A_{23}\end{bmatrix}\begin{bmatrix}\underline{x}_m(t_i)\\ \underline{u}_m(t_i)\\ \underline{n}_d(t_i)\end{bmatrix} \qquad (2\text{-}44)$$

2-4.1 Open-Loop CGT

To solve this open loop CGT problem, the equations that the A_{ij} matrices in Equation (2-44) must satisfy have to be set up and solved. By augmenting the forward difference expression for $\underline{x}_I(t_i)$ from Equation (2-42a) with $\underline{y}_I(t_i)$ from Equation (2-42b):

$$\begin{bmatrix} \underline{x}_I(t_{i+1}) - \underline{x}_I(t_i) \\ \underline{y}_I(t_i) \end{bmatrix} = \begin{bmatrix} (\Phi-I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} \underline{x}_I(t_i) \\ \underline{u}_I(t_i) \end{bmatrix} + \begin{bmatrix} E_x \\ E_y \end{bmatrix} \underline{n}_d(t_i) \quad (2\text{-}45)$$

By substituting the assumed forms of $\underline{x}_I(t_i)$ and $\underline{u}_I(t_i)$ from Equation (2-44), this becomes:

$$\begin{bmatrix} \underline{x}_I(t_{i+1}) - \underline{x}_I(t_i) \\ \underline{y}_I(t_i) \end{bmatrix} = \begin{bmatrix} (\Phi-I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} + \begin{bmatrix} E_x \\ E_y \end{bmatrix} \underline{n}_d(t_i) \quad (2\text{-}46)$$

The forward difference expression for $\underline{x}_I(t_i)$ can also be obtained by writing the upper partition of Equation (2-44) for times t_i and t_{i+1} and taking the difference to yield:

$$[\underline{x}_I(t_{i+1})-\underline{x}_I(t_i)] = [A_{11} \; A_{12} \; A_{13}] \begin{bmatrix} \underline{x}_m(t_{i+1})-\underline{x}_m(t_i) \\ \underline{u}_m(t_{i+1})-\underline{u}_m(t_i) \\ \underline{n}_d(t_{i+1})-\underline{n}_d(t_i) \end{bmatrix} \quad (2\text{-}47)$$

Assuming $\underline{u}_m$ to be either constant or at least slowly varying with respect to the sample period (i.e., $\underline{u}_m(t_{i+1})-\underline{u}_m(t_i) \approx \underline{0}$), using the state models for $\underline{x}_m$ and $\underline{n}_d$ as given in Equations (2-36) and (2-40) respectively, and deleting the driving noises yields:

$$[\underline{x}_I(t_{i+1})-\underline{x}_I(t_i)] =$$

$$= [A_{11} \; A_{12} \; A_{13}] \begin{bmatrix} (\Phi_m-I) & B_{dm} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & (\Phi_n-I) \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} \quad (2\text{-}48)$$

Using the fact that $\underline{y}_I=\underline{y}_m$ for all time t_i, one can write the expression

$$\underline{y}_I(t_i) = [C_m \; D_m] \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \end{bmatrix} \quad (2\text{-}49)$$

Multiplying the matrices of Equation (2-48) gives

$$\begin{bmatrix} \underline{x}_I(t_{i+1})-\underline{x}_I(t_i) \\ \underline{y}_I(t_i) \end{bmatrix} =$$

$$= \begin{bmatrix} A_{11}(\mathbf{E}_m-I) & A_{11}B_{dm} & A_{13}(\mathbf{E}_n-I) \\ C_m & D_m & 0 \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} \quad (2\text{-}50)$$

Equating the two expressions for the forward difference of $\underline{x}_I(t_i)$ and the output $\underline{y}_I(t_i)$, Equations (2-46) and (2-50), yields:

$$\begin{bmatrix} (\mathbf{E}-I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} + \begin{bmatrix} E_x \\ E_y \end{bmatrix} \underline{n}_d(t_i) =$$

$$= \begin{bmatrix} A_{11}(\mathbf{E}_m-I) & A_{11}B_{dm} & A_{13}(\mathbf{E}_n-I) \\ C_m & D_m & 0 \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} \quad (2\text{-}51)$$

which can be rearranged into:

$$\begin{bmatrix} (\mathbf{E}-I) & B_d \\ C & D_y \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix}$$

$$- \begin{bmatrix} A_{11}(\mathbf{E}_m-I) & A_{11}B_{dm} & A_{13}(\mathbf{E}_n-I)-E_x \\ C_m & D_m & -E_y \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i) \\ \underline{u}_m(t_i) \\ \underline{n}_d(t_i) \end{bmatrix} = \underline{0} \quad (2\text{-}52)$$

Since this expression must hold true for arbitrary $\underline{x}_m$, $\underline{u}_m$, and $\underline{n}_d$ at any given sample time, the quantity in the braces must equal the zero matrix. Therefore,

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} =$$

$$= \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} \begin{bmatrix} A_{11}(\mathbf{E}_m-I) & A_{11}B_{dm} & A_{13}(\mathbf{E}_n-I)-E_x \\ C_m & D_m & -E_y \end{bmatrix} \quad (2\text{-}53)$$

which can be written as the following set of partitioned equations:

$$A_{11} = \pi_{11}A_{11}(\mathbf{E}_m-I) + \pi_{12}C_m \quad (2\text{-}54a)$$

$$A_{12} = \pi_{11}A_{11}B_{dm} + \pi_{12}D_m \quad (2\text{-}54b)$$

$$A_{13} = \pi_{11}A_{13}(\mathbf{E}_n-I) - \pi_{11}E_x - \pi_{12}E_y \quad (2\text{-}54c)$$

$$A_{21} = w_{21}A_{11}(\Phi_m - I) + w_{22}C_m \quad (2\text{-}54d)$$

$$A_{22} = w_{21}A_{11}B_{dm} + w_{22}D_m \quad (2\text{-}54e)$$

$$A_{23} = w_{21}A_{13}(\Phi_n - I) - w_{21}E_x - w_{22}E_y \quad (2\text{-}54f)$$

Equations (2-54a) and (2-54c) are of the form X = AXB + C, assuming Φ_m and/or Φ_n are not the identity, and can be solved numerically for X [4]. Achieving a unique solution by the above method is dependent upon the following relationships. First, the product of any of the eigenvalues of w_{11} and any eigenvalue of $(\Phi_m - I)$ must not be unity and second, the product of any of the eigenvalues of w_{11} and $(\Phi_n - I)$ also cannot equal one.

Once A_{11} and A_{13} are determined, the rest of the A_{ij} matrices can be evaluated and the open-loop CGT law can be written from the lower partition of Equation (2-44) as [11]:

$$\underline{u}_I(t_i) = A_{21}\underline{x}_m(t_i) + A_{22}\underline{u}_m(t_i) + A_{23}\underline{n}_d(t_i) \quad (2\text{-}55)$$

The open-loop CGT is shown in Figure 2-3.

2-4.2 CGT/Regulator

The open-loop CGT law is usually not appropriate as it can not be used with unstable plants or plants with undesirable performance, and it cannot handle the uncertain parameters or unmodeled disturbances that generally occur in the true system. Therefore, to force the perturbations from the ideal plant trajectory to zero, a feedback controller must be developed. The first step in this process is to assume that the open-loop CGT law, $\underline{u}_I(t_i)$ has

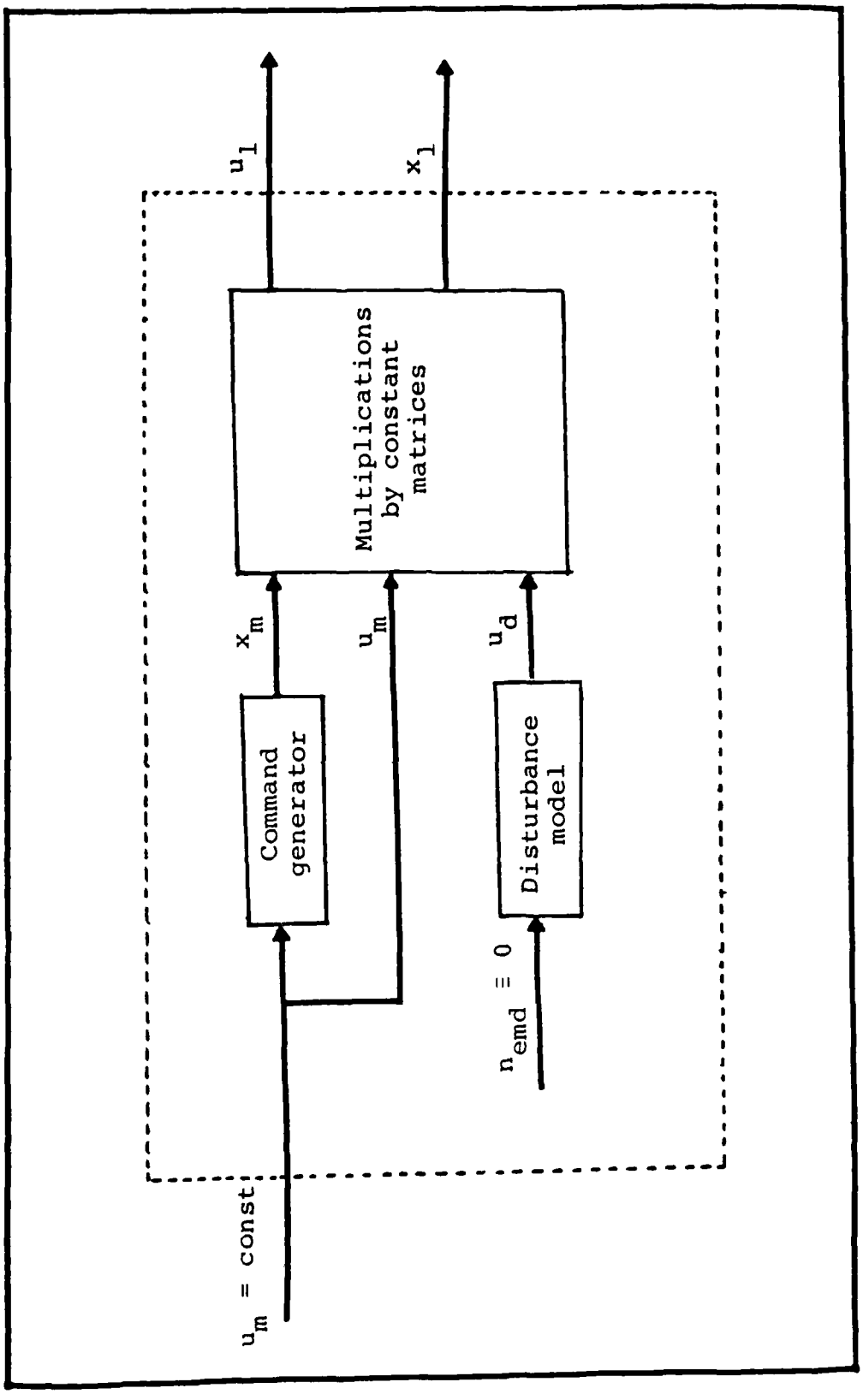


Figure 2-3. Open-Loop CGT Controller

been evaluated for all t_i, yielding the ideal state trajectory $\underline{x}_I(t_i)$ for all t_i.

To generate the LQ optimal regulator for the deterministic system, or by certainty equivalence, the LQG regulator for a full-state feedback sytem, perturbation variables must be defined:

$$\delta\underline{x}(t_i) \equiv \underline{x}(t_i) - \underline{x}_I(t_i) \tag{2-56a}$$

$$\delta\underline{u}(t_i) \equiv \underline{u}(t_i) - \underline{u}_I(t_i) \tag{2-56b}$$

$$\delta\underline{y}_C(t_i) \equiv \underline{y}_C(t_i) - \underline{y}_I(t_i) = \underline{y}_C(t_i) - \underline{y}_m(t_i) \tag{2-56c}$$

This yields a deterministic perturbation state equation of

$$\delta\underline{x}(t_{i+1}) = \Phi_d\delta\underline{x}(t_i) + B_d\delta\underline{u}(t_i) \tag{2-57}$$

and a steady state constant-gain LQ optimal regulator law as

$$\delta\underline{u}(t_i) = -G_C^*\delta\underline{x}(t_i) \tag{2-58}$$

This can be written in the preferred incremental form to yield an equivalent closed-loop CGT control law [20] of

$$\begin{aligned}\underline{u}(t_{i+1}) &= \underline{u}(t_i) + [\underline{u}_I(t_{i+1}) - \underline{u}_I(t_i)] \\ &+ G_C^*[\underline{x}_I(t_{i+1}) - \underline{x}_I(t_i)] + G_C^*[\underline{x}(t_{i+1}) - \underline{x}(t_i)]\end{aligned} \tag{2-59}$$

2-4.3 Closed-Loop Command Generator Tracking with PI Control

The derivations of Sections 2-3.2 and 2-4.1 may now be combined in a closed-loop CGT/PI controller. This controller will force the actual system's mean output to match that of the command model in steady state, even with some modeling errors; it can handle constant unmodeled disturbances as well as the modeled ones; and it includes the model control's feedforward contribution to the real

system control signal.

In the previous sections, it has been assumed that $\underline{u}_m$ is constant from t_o forward. In reality $\underline{u}_m$ will change and this will conceptually cause the time index to be set to zero. This causes an inconsistency in the definition of the ideal trajectory. To avoid this, the actual implementation of the command generator model to take the form [15]:

$$\underline{x}_m(t_{i+1}) = \Phi_m \underline{x}_m(t_i) + B_{dm}\underline{u}_m(t_{i+1}) \qquad (2\text{-}60)$$

as opposed to the form of Equation (2-36).

Using the open-loop CGT law $\underline{u}_I(t_i)$ of Equation (2-55), the associated ideal state trajectory $\underline{x}_I(t_i)$ of Equation (2-42a), and defining the ideal trajectory of control differences as:

$$\begin{aligned}\Delta\underline{u}_I(t_i) &\equiv \underline{u}_I(t_{i+1}) - \underline{u}_I(t_i) \\ &= A_{21}[\underline{x}_m(t_{i+1}) - \underline{x}_m(t_i)] + A_{23}[\underline{n}_d(t_{i+1}) - \underline{n}_d(t_i)] \qquad (2\text{-}61)\end{aligned}$$

The deterministic augmented perturbation system state equation is similar to Equation (2-25), but with $\Delta\underline{u}$ replaced by $\delta\Delta\underline{u}$, where $\delta\Delta\underline{u}$ is defined by

$$\delta\Delta\underline{u}(t_i) \equiv \Delta\underline{u}(t_i) - \Delta\underline{u}_I(t_i) \qquad (2\text{-}62)$$

The LQ optimal perturbation regulator solution to this system is given by:

$$\delta\Delta\underline{u}(t_i) = -G_{c1}{}^{*}\delta\underline{x}(t_i) - G_{c2}{}^{*}\delta\underline{u}(t_i) \qquad (2\text{-}63)$$

Substituting for $\delta\Delta\underline{u}$, $\delta\underline{x}$, and $\delta\underline{u}$; and shifting the time argument backward by one sample period yields:

$$\underline{u}(t_i) = \underline{u}(t_{i-1}) + A_{21}[\underline{x}_m(t_i)-\underline{x}_m(t_{i-1})] +$$
$$+ A_{23}[\underline{n}_d(t_i)-\underline{n}_d(t_{i-1})] -$$
$$- G_{c1}^*[\underline{x}(t_{i-1})-A_{11}\underline{x}_m(t_{i-1})-A_{12}\underline{u}_m(t_i)-A_{13}\underline{n}_d(t_{i-1})] -$$
$$- G_{c2}^*[\underline{u}(t_{i-1})-A_{21}\underline{x}_m(t_{i-1})-A_{22}\underline{u}_m(t_i)-A_{23}\underline{n}_d(t_{i-1})] \quad (2\text{-}64)$$

which, while being in the incremental form, is only a type-0 controller. Note that the time arguments on both $\underline{u}_m$'s reflect the modification of Equation (2-60)

The desired integral property can be achieved by a controller of the form:

$$\underline{\sigma u}(t_i) = \underline{\sigma u}(t_{i-1}) - [K_x \; K_z] \begin{bmatrix} \underline{\sigma x}(t_i) - \underline{\sigma x}(t_{i-1}) \\ \underline{\sigma y}_c(t_{i-1}) \end{bmatrix} \quad (2\text{-}65)$$

Using the perturbation variables defined in the previous section, Equation (2-65) can be written as:

$$\underline{u}(t_i) = \underline{u}_I(t_i) + \underline{u}(t_{i-1}) - \underline{u}_I(t_{i-1})$$
$$- K_x[\underline{x}(t_i) - \underline{x}(t_{i-1})] + K_x[\underline{x}_I(t_i)-\underline{x}_I(t_{i-1})]$$
$$+ K_z[\underline{y}_m(t_{i-1})-\underline{y}_c(t_{i-1})] \quad (2\text{-}66)$$

Writing the backward difference of Equation (2-44) yields:

$$\begin{bmatrix} \underline{x}_I(t_i)-\underline{x}_I(t_{i-1}) \\ \underline{u}_I(t_i)-\underline{u}_I(t_{i-1}) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \begin{bmatrix} \underline{x}_m(t_i)-\underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i)-\underline{u}_m(t_{i-1}) \\ \underline{n}_d(t_i)-\underline{n}_d(t_{i-1}) \end{bmatrix} \quad (2\text{-}67)$$

This can be substituted into Equation (2-66) to yield the desired incremental form of the CGT/PI controller [20]:

$$\underline{u}(t_i) = \underline{u}(t_{i-1}) - K_x[\underline{x}(t_i)-\underline{x}(t_{i-1})]$$
$$+ K_z [C_m \; D_m] \begin{matrix} \underline{x}_m(t_{i-1}) \\ \underline{u}_m(t_i) \end{matrix} - [C \; D_y] \begin{matrix} \underline{x}(t_{i-1}) \\ \underline{u}(t_{i-1}) \end{matrix}$$
$$+ [K_x A_{11}+A_{21}] [\underline{x}_m(t_i)-\underline{x}_m(t_{i-1})]$$
$$+ [K_x A_{13}+A_{23}] [\underline{n}_d(t_i)-\underline{n}_d(t_{i-1})] \quad (2\text{-}68)$$

which is shown in Figure 2-4.

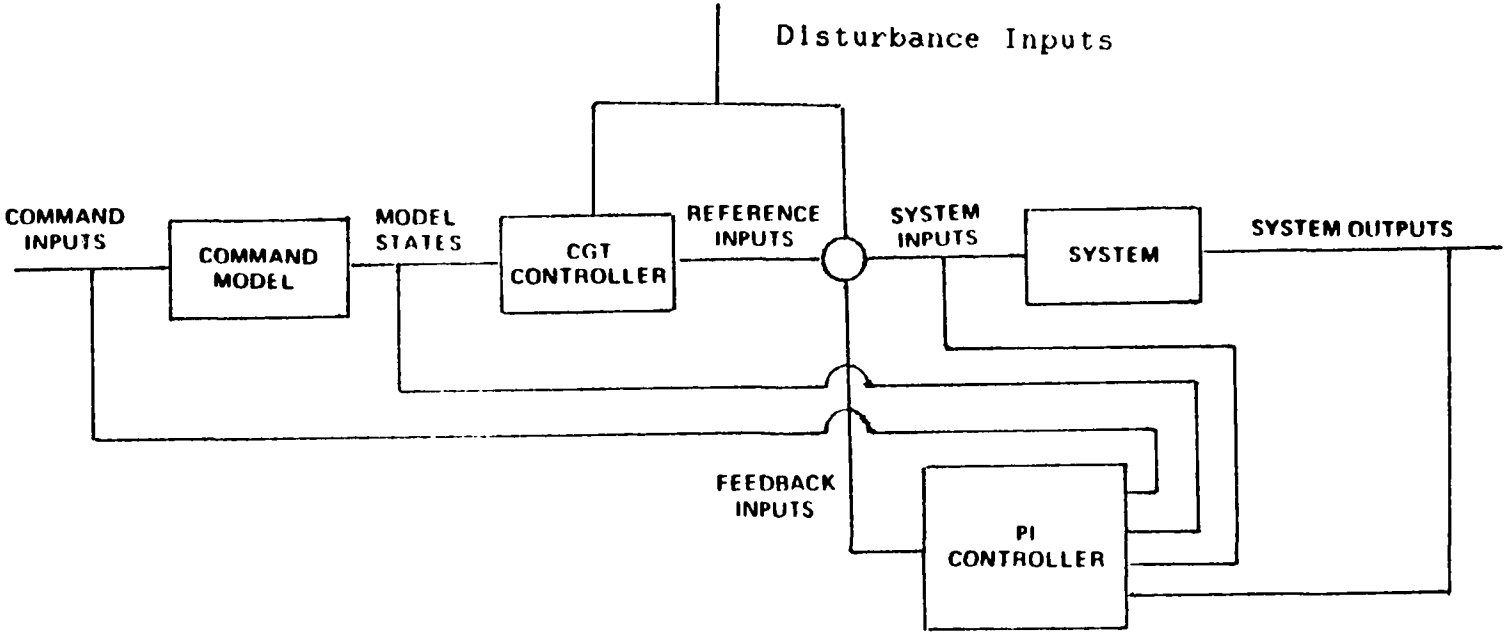


Figure 2-4. CGT/PI Controller

2-4.4 Model Following Controllers

The CGT/PI controller derived in the above section is an example of a model following controller. The CGT is an explicit- model following controller, in which the command model used is the ideal performance model, possibly including a disturbance rejection model. The CGT and the explicit-model follower are feedforward phenomena and have no effect on system closed-loop properties [20].

Implicit model-following control may be added to the PI regulator by the inclusion of an implicit model into the performance index. The PI controller forces the perturbation outputs of the system to mimic the dynamics of the implicit model. The addition of the implicit model incorporates weights on deviations in the rate of change of the output variables from the desired characteristics and also on the control pseudo-rates. As the implicit model-following controller is a feedback phenomenon, it directly affects the closed-loop characteristics of the system, including stability robustness [11,20,21].

The following derivation of the implicit model following controller is for the continuous-time full-state feedback regulator, but the concepts have been extended to sampled-data PI controllers [11,24]. Consider a full-state feedback system adequately described as the linear time-invariant model:

$$\dot{\underline{x}}(t) = F\,\underline{x}(t) + B\,\underline{u}(t) \qquad (2\text{-}69)$$

with a quadratic cost function defined as:

$$J_I = \tfrac{1}{2}\int_0^\infty [\underline{x}^T(t)X\underline{x}(t) + \underline{u}^T(t)U\underline{u}(t)]\,dt \qquad (2\text{-}70)$$

which places a quadratic penalty on deviations from zero of both state trajectory and control energy. Given the infinite time interval of Equation (2-70), a constant-gain full-state feedback control law, derived using LQ synthesis techniques, would take the form:

$$\underline{u}^*(t) = -G_C^*\underline{x}(t) \qquad (2\text{-}71)$$

or, for the case of an equivalent discrete-time system as shown in Section 2-2, the law would take the form of Equation (2-6), as defined by Equations (2-7) through (2-9).

It is assumed that the system outputs can be written as:

$$\underline{y}_C(t) = C\underline{x}(t) \qquad (2\text{-}72)$$

a linear combination of the states. If the desire is not to drive the components of the state vector to zero, but to force the system output to match the dynamics of a given model system depicted as:

$$\dot{\underline{y}}_m(t) = F_m\underline{y}_m(t) \qquad (2\text{-}73)$$

the cost in Equation (2-70) associated with the state deviations would change from $\underline{x}^TX\underline{x}$ to $(\dot{\underline{y}}-F_m\underline{y})^TY_I(\dot{\underline{y}}-F_m\underline{y})$. By combining this relation with Equations (2-69) and (2-72), the cost function may be rewritten as:

$$J_I = \tfrac{1}{2}\int_0^\infty [\underline{x}^TX_I\underline{x} + \underline{u}^TU\underline{u} + 2\underline{x}^TS_I\underline{u}]\,dt \qquad (2\text{-}74a)$$

where

$$X_I = (CF - F_mC)^TY_I(CF - F_mC) \qquad (2\text{-}74b)$$

$$U_I = U + B^T C^T Y_I C B \qquad (2\text{-}74c)$$

$$S_I = (CF - F_m C)^T Y_I C B \qquad (2\text{-}74d)$$

As with the S matrix of Equation (2-5), the purpose of the S_I terms in Equation (2-74a) is to exert control on the rates of change of system outputs. Given this cost function, the control law in either Equation (2-71) or (2-6) can be synthesized by LQ techniques [21,24].

At this stage in the development of a flight control law, the goal is to design as robust a controller as is possible, to set a bound on performance that, once a Kalman filter is embedded in the loop, will be asymptotically approached using Loop Transfer Recovery (LTR) robustness enhancement techniques. It has been shown that, to decrease the sensitivity of the controller to parameter variations, and therefore, make it as robust as possible, the closed-loop eigenvectors should be kept as orthogonal as possible [12,24]. This may be accomplished by the proper selection of the implicit model and the weights on the control pseudo-rates and on deviations of output derivatives from model derivatives. With the controller designed for robustness, the technique to be discussed in the next chapter may be used to recover some of the robustness lost from the inclusion of a Kalman filter.

2-5 SUMMARY

This chapter developed the deterministic optimal LQ controllers for continuous-time systems having sampled-data measurements. The development began with a simple regulator, and then a proportional plus integral controller was described. Next, a command generator tracker was developed and was subsequently combined with the regulator and the PI controller, respectively. Finally, model following controllers were discussed, along with their contribution to system robustness. In the next chapter uncertainty and incomplete measurements (versus full and perfect access to the states) will be introduced, requiring the addition of the Kalman Filter.

III. KALMAN FILTERING AND ROBUSTNESS

3-1 Introduction

In the previous chapter, controller development was based implicitly on the assumption that all of the states of the assumed system model were accessible and measured perfectly. This is obviously not the general case which involves incomplete, noise-corrupted outputs of physical sensors. The Kalman filter is often used, and is developed here, to generate optimal estimates of the states from the outputs of these sensors [22]. By certainty equivalence [20], this structure then provides the optimal stochastic controller under the LQG assumptions.

The addition of the Kalman filter into the loop has a negative effect on the robustness of the system, i.e. the ability of the system to tolerate design uncertainties while providing stable characteristics and desired performance. This chapter will address a specific form of Kalman filter tuning as a method of asymptotically recovering the good robustness characteristics achieved for full-state feedback controllers via implicit model following in the previous chapter.

3-2 Kalman Filtering

The CGT/PI controller of Chapter 2 assumes full knowledge of all system and disturbance state vectors. In general the states are not entirely available directly from measurements of the states, but are only available as

sensor measurements, often incomplete (in the sense that there are not separate measurements of all states of interest), and corrupted by noise. This motivates the use of a Kalman filter to generate estimates of the states. A development of the Kalman filter follows, but for a more in-depth explanation, see Reference 19.

The available sampled-data sensor outputs are assumed to be modelled by:

$$\underline{z}(t_i) = H\underline{x}(t_i) + \underline{v}(t_i) \tag{3-1}$$

where H is the measurement matrix and $\underline{v}(t_i)$ is a zero-mean white Gaussian noise with

$$E\{\underline{v}(t_i)\underline{v}^T(_j)\} = R\,\delta_{ij} \tag{3-2}$$

This measurement noise is assumed to be independent of the dynamics driving noise.

Through the use of Bayes rule, the optimal estimation algorithm (Kalman filter) can be defined in terms of the conditional mean and covariance of the state vector just after a measurement has been incorporated :

$$\hat{\underline{x}}(t_{i-1}^+) \equiv E\{\underline{x}(t_{i-1}) | Z(t_{i-1}) = Z_{i-1}\} \tag{3-3}$$

$$P(t_{i-1}^+) \equiv E\{[\underline{x}(t_{i-1}) - \hat{\underline{x}}(t_{i-1}^+)][\underline{x}(t_{i-1}) - \hat{\underline{x}}(t_{i-1}^+)]^T | Z(t_{i-1}) = Z_{i-1}\} \tag{3-4}$$

where $Z(t_{i-1})$ is a composite vector of the entire measurement history and Z_{i-1} is a vector of the realizations of $Z(t_{i-1})$. These values are propagated to the next measurement time to yield :

$$\hat{\underline{x}}(t_i^-) \equiv E\{\underline{x}(t_i) | Z(t_{i-1}) = Z_{i-1}\} \tag{3-5}$$

$$P(t_i^-) \equiv E\{[\underline{x}(t_i)-\hat{\underline{x}}(t_i^-)][\underline{x}(t_i)-\hat{\underline{x}}(t_i^-)]^T \mid Z(t_{i-1})=Z_{i-1}\} \quad (3\text{-}6)$$

Using these relationships and the system equation from Chapter 2, the Kalman filter equations are as follows:

$$\hat{\underline{x}}(t_i^-) = \Phi\, \hat{\underline{x}}(t_{i-1}^+) + B_d\, \underline{u}(t_{i-1}) \quad (3\text{-}7a)$$

$$P(t_i^-) = \Phi\, P(t_{i-1}^+)\, \Phi^T + G_d\, Q_d(t_{i-1})\, G_d^T \quad (3\text{-}7b)$$

$$K(t_i) = P(t_i^-)\, H^T\, [H\, P(t_i^-)\, H^T + R]^{-1} \quad (3\text{-}7c)$$

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + K(t_i)\, [\underline{z}_i - H\, \hat{\underline{x}}(t_i^-)] \quad (3\text{-}7d)$$

$$P(t_i^+) = P(t_i^-) - K(t_i)\, H\, P(t_i^-) \quad (3\text{-}7e)$$

The Kalman filter is inserted into the loop as shown in Figure 3-1. The outputs of the filter are fed into the optimal deterministic controller, replacing the actual state $\underline{x}(t_i)$ that was assumed accessible during that controller's derivation, which yields the control vector $\underline{u}(t_i)$.

3.3 Effects of Kalman Filtering on Robustness

Robustness is the ability of a system to maintain stability and/or performance in the presence of variations from design conditions [26]. Full-state feedback controllers designed via LQ technique, while varying in the degree of robustness, all attain at least a minimum guaranteed stability robustness. When the states of the controller's system are estimated by a Kalman filter, much of the robustness and all of the guarantees are eliminated. The desire in multivariable flight control design is to achieve a controller that will provide both stability and

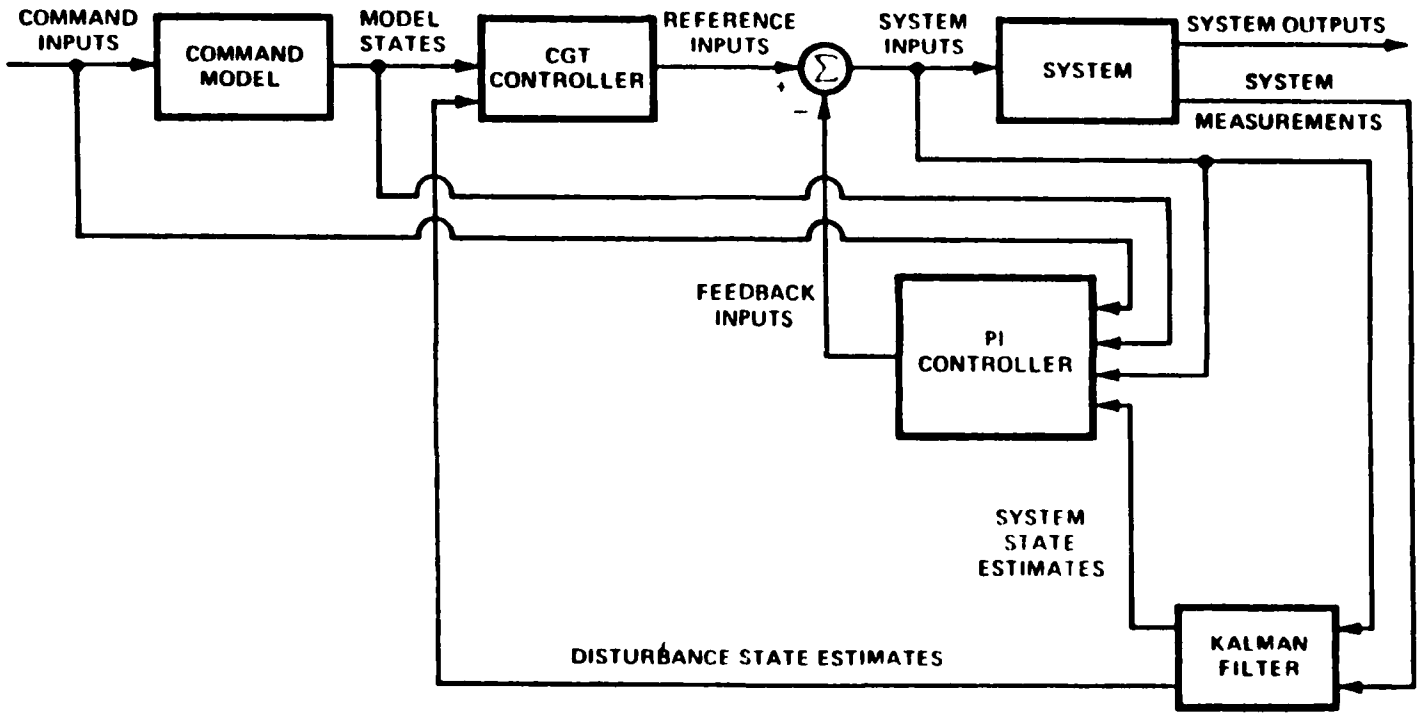


Figure 3-1. CGT/PI/KF Controller

performance robustness in the face of uncertainties in the system, and at least stability robustness in the presence of massive changes in system characteristics [21]. These changes may result from, for instance, mismodelling of the system, ignored actuator dynamics, nonlinearities, or failure of control surfaces.

3-4 Robustness Analysis

In this research, robustness analysis was conducted by using proposed controllers in simulated, linear and nonlinear "truth-model" environments. Figure 3-2 shows the basic form of these simulations, where the truth model differs in structure between the linear and nonlinear simulations, but the controller does not. The linear simulation was a covariance analysis using the program PFEVAL written by Lt. A. Moseley for his AFIT thesis [25]. The results of this analysis are the root mean square (rms) values of the state values and control inputs. For a linear system, the covariance analysis is preferred as a single run of the covariance analysis yields the statistics required for controller evaluation, but, in the face of nonlinearities, a Monte Carlo study is required [19]. This was the purpose of the nonlinear simulation which allowed for the saturation of actuators, as well as unmodelled higher-order dynamics, and also variation in the parameters that were used in the design of the controller.

The tool used for the nonlinear simulation was

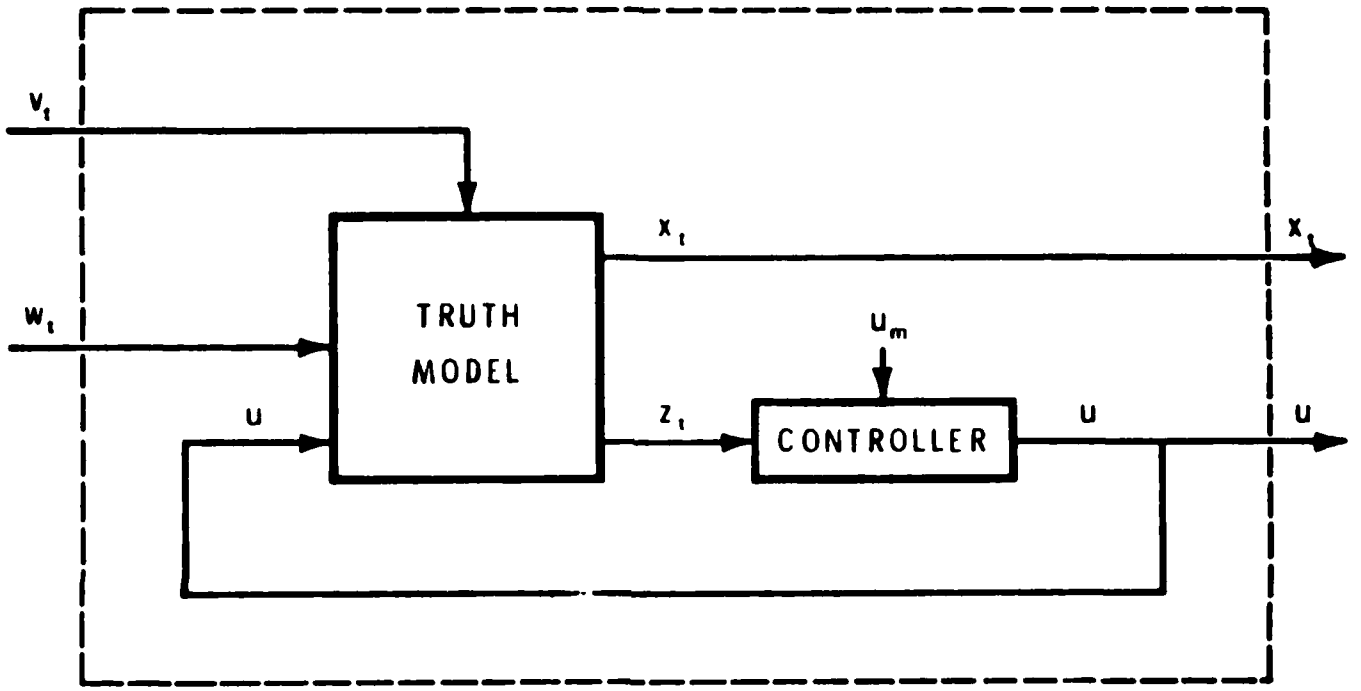


Figure 3-2. Performance Evaluation

originally developed by Major William Miller [24] and was modified by Lt. Robert Houston and this author. Originally, the program ODEACT was hardwired for the AFTI F-16 and did not allow for the inclusion of a Kalman filter in the evaluation. The modifications to the program, renamed ODEF15, include hardwiring of the program for the STOL F-15 and possible inclusion of a Kalman filter with Monte Carlo analysis. The actual modifications and a list of the computer code are included in Appendix A.

3-5 Robustness Improvements

Two methods are used in this research to improve the robustness of the controlled system. The first, implicit model following, is involved directly in the design of the deterministic optimal full-state feedback controller and was discussed in Chapter 2. The second is the Doyle and Stein technique for tuning the Kalman filter to recover robustness characteristics of a full-state feedback design.

In their paper "Robustness with Observers" [9], J. C. Doyle and Gunter Stein develop a technique for robustifying a controlled system that is fed estimates of states from an observer or state estimator. Assumptions of controllability, observability, and minimum phase of the system design model are made and only continuous time systems are considered. The basic claim for this technique is that, if the mappings of the return difference matrices (the multiple-input, multiple-output analog to the

denominator of the single-input, single-output closed-loop transfer function) of the observer-based controller are asymptotically equal to those of the full-state feedback controller when the control loops are opened at the point of entry of the control inputs, then the robustness of the controller will asymptotically approach that of the full-state feedback controller.

Previous AFIT papers [15,17,21] have extended the Doyle and Stein technique to discrete-time measurements and have attempted to use this method in conjunction with a PI controller. The technique was successfully applied to a discrete-time regulator, but could not beneficially be implemented with a PI controller. The discussion of the discrete-time methods below involves an approximation to the continuous-time case and these mehtods are set up for robustness enhancement of regulators as opposed to PI controllers.

To implement the Doyle and Stein technique, the $G_d Q_d G_d^T$ matrix of Equation (3-7b) is modified such that G_d is now set to the identity matrix, I, and:

$$Q_d(q) = Q_{do} + q^2 B_d V B_d^T \qquad (3\text{-}8)$$

where Q_{do} is the original noise covariance as was defined in Equation (2-4), q is a scalar variable that is a function of the amount of robustness enhancement desired, and V is a positive definite symmetric matrix, often set to the identity matrix, I. This can be interpreted as adding a noise of strength q^2V at the control entry points. The

modified $G_dQ_dG_d^T$ is now inserted into the Kalman filter equations. The resultant controller formed, as a cascade of a PI controller and the Kalman filter, should be more robust than the same type controller without the purposeful retuning of the filter. Note that when q is zero, the $G_dQ_dG_d^T$ is unchanged and that, for large (but finite) q, Equation (3-8) might be replaced with $q^2B_dVB_d^T$ alone.

3-6 Summary

This chapter briefly introduced the Kalman filter and discussed the negative effects of inclusion of the filter on robustness. A methodology was introduced that consists of developing a highly robust full-state feedback law by the implicit model following techniques of the previous chapter, and then recovering much of the robustness once a filter is inserted into the loop by a specific form of filter tuning.

In the next chapter, the methods of performance analysis will be presented.

IV. F-15 FLIGHT CONTROL DESIGN

4-1 Introduction

In this chapter the dynamic equations for the Short Takeoff and Landing (STOL) F-15 will be presented and used to assemble a truth model and the reduced order models required for flight control design. To allow for the use of battle damaged runways, an aircraft must have STOL capability. To this end, it has been proposed that two-dimensional nozzles be retro-fitted to an F-15 along with the addition of canards. The addition of the thrust vectoring nozzles and control surfaces provides an extra degree of freedom to the control engineer in his design of the control laws for the aircraft. This allows him to develop a control system that will minimize the approach velocity and provide precision touchdown capability.

The design of a flight controller begins with the aircraft nonlinear equations of motion, which are linearized about specified points in the flight envelope and thus provide a number of design models which are adequately characterized as linear and time-invariant. The data provided for the design was from the McDonnell Douglas Aircraft Corporation (McAir) through AFWAL/FIGX [23].

4-2 Reduction of Aerodynamic Data

The data obtained from McAir was in the form of raw aerodynamic data. This data was entered into a version of the Conversion and Transformation Program (CAT), developed

originally by Mr. A. Finley Barfield [3] for the AFTI F-16 aircraft and modified by Capt. Greg Mandt, Lt. Bruce Clough, and this author to conform to the control surfaces and dimensionality available with the STOL F-15. A listing of the software code, now called STOLCAT, and a representative run of the program are given in Appendix B. This program takes the basic aerodynamic and aircraft data and forms a state-space representation for the system for the point about which the equations of motion have been linearized.

For the design of a robust controller for the approach and landing phase, the flight condition of 119 knots (200 feet per second), at sea level, with an aircraft gross weight of 33,576 pounds was chosen as the equilibrium. For all controller design and robustness checks, it was assumed that the aircraft was known with probability one to be in the appropriate equilibrium flight condition at time t_0. This point is especially important for the covariance analysis as it establishes the initial values for the covariance matrix. It also gives the initial conditions for the Monte Carlo studies.

For robustness studies, it was necessary to obtain the linear perturbation models for other points in the flight envelope. These models are used as truth models in the software employed to evaluate the controller design and are shown in Appendix C.

4-3 Design Model

In the design of a controller for the specified flight condition, modifications were required to the state-space representation acquired from STOLCAT, as the output from STOLCAT provides inputs from each control surface to each state vector element. These modifications involved achieving the proper dimensions for the control input matrix and specifying the output matrix. Due to software limitations, the number of inputs must equal the number of outputs, and also due to the desire to keep the results of this study comparable to thesis efforts on the same problem using Qualitative Feedback Theory [7] and the Porter method [1], a three-input/three-output system is used. This is a restriction on design latitude that need not be levied on the LQG design approach to this problem. The expected result of this restriction is a sub-optimal controller when compared to one designed using a greater number of inputs and outputs (as is seen in Appendix D).

The appropriate output variables for an approach and landing for a STOL aircraft, or any aircraft where landing roll-out is a critical factor, are flight path and velocity. Previous research has determined that controlling the pitch rate along with flight path angle and velocity, yields a better controller [4,5]. In this design, these output variables are to be controlled by the canards, (tied aerodynamically to the flaps and ailerons to limit the control inputs), the stabilator, and the reverser

vanes in the engines. To reduce effects of nonlinearities and to simplify the analysis of the controller, the combination of the canards with the flaps and ailerons is done so that an input that would saturate a control surface would saturate all six surfaces at the same time. This was accomplished by scaling the appropriate entries in the control input matrix. The procedure for this combination and scaling are shown in Appendix E.

The resulting design model is a four-state, time invariant model of the form

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} + \underline{w} \qquad (4\text{-}1)$$

where the state vector $\underline{x}$ has components

x_1 = velocity (feet per second)

x_2 = pitch rate (radians per second)

x_3 = pitch angle (radians)

x_4 = angle of attack (radians)

and the control inputs are:

u_1 = canard/flap/aileron deflection

u_2 = stabilator deflection

u_3 = reverser vane deflection (4-2)

and the elements of the noise vector $\underline{w}$ are the noises affecting the associated state variables. The system matrices are

$$A = \begin{bmatrix} -6.9009 \times 10^{-2} & -40.16 & 0.3352 & -31.54 \\ -3.6036 \times 10^{-4} & -0.9913 & 1.367 & 0 \\ 9.2254 \times 10^{-6} & 0.9796 & -0.6392 & -3.233 \times 10^{-2} \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (4\text{-}3)$$

$$B = \begin{bmatrix} -1.142 & -3.238 & -21.86 \\ 0.8710 & -1.578 & -2.507 \times 10^{-2} \\ -5.209 \times 10^{-2} & -7.6686 \times 10^{-2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4\text{-}4)$$

with Q, the strength of the noise $\underline{w}$, set initially to zero. The output equation defining the variables over which control is specifically sought, is

$$\underline{y}_C = C\underline{x} \quad (4\text{-}5)$$

here

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \quad (4\text{-}6)$$

which yields the outputs of velocity, pitch rate, and flight path angle. For this research, it is assumed that noisy measurements of all of the elements of the controlled variable vector are available. Therefore, in the measurement equation

$$\underline{z} = H\underline{x} + \underline{v} \quad (4\text{-}7)$$

where the measurement matrix H is equal to the ouput matrix C and the strength of the zero-mean, white Gaussian noise vector $\underline{v}$ is

$$R = \begin{bmatrix} 4.76 \times 10^{-6} & 0 & 0 \\ 0 & 1.22 \times 10^{-5} & 0 \\ 0 & 0 & 3.22 \times 10^{-5} \end{bmatrix} \quad (4\text{-}8)$$

As sensor noise data for the STOL F-15 was not available, the values above were taken from Reference 11.

This model was input into the computer aided design tool developed by Floyd [11] and Moseley [25] and as modified by Miller [24]. It served as the basis for the controller design and the initial evaluation of the controller, as this tool could only evaluate the full-state feedback controller or the filter against the truth model, but not the combination of the controller and the filter.

4-4 Explicit Command Model

The function of the explicit command model in this application is to specify the desired handling qualities for the aircraft. From Mil-F-8785C [2], the short period response in the landing and approach phase should exhibit a damping ratio between .35 and 1.3 with a frequency range, for the angle of attack values involved, of .87 to 4.1 radians per second. These values are incorporated directly into the explicit command model along with the desire to command a zero net change in velocity and pitch rate. Therefore the command model takes the form

$$\dot{\underline{x}}_m = A_m \underline{x}_m + B_m \underline{u}_m \quad (4\text{-}9)$$

with

x_{m1} = commanded velocity

x_{m2} = commanded pitch rate

x_{m3} = commanded flight path angle

x_{m4} = commanded rate of change of flight path angle

and with u_m equal to a unit step used to define the desired second- order response in flight path angle. In this model, A_m and B_m are:

$$A_m = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -16 & -5.6 \end{bmatrix} \quad (4\text{-}10)$$

$$B_m = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 16 \end{bmatrix} \quad (4\text{-}11)$$

The commanded output variables for this application are the velocity, the pitch rate, and flight path angle, with velocity and pitch rate commanded to zero and the flight path angle to follow the handling qualities characterics specified above. This yields an output equation of

$$\underline{Y}_m = C_m \underline{X}_m \quad (4\text{-}12)$$

and

$$C_m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4\text{-}13)$$

4-5 Implicit Command Model

As was discussed in Chapter 3, the purpose of the implicit model is to enhance the robustness characteristics of the controller and, to accomplish this, the eigenvectors of the closed-loop system must be kept as orthogonal as possible. The implicit control command model obeys the same form of state and output equations as the explicit command model, but with $\underline{x}_m$ set equal to the outputs of the design model, namely velocity, pitch rate, and flight path angle, and

$$A_m = \begin{bmatrix} -M_1 & 0 & 0 \\ 0 & -M_2 & 0 \\ 0 & 0 & -M_3 \end{bmatrix} \qquad (4\text{-}14)$$

and B_m, C_m, and D_{ym} are set to zero as they have no effect upon the implicit model. Note that the choice of a diagonal A_m yields orthogonal eigenvectors. The values for M_1, M_2, and M_3 are determined iteratively and have a direct consequence on robustness. This iteration process will be seen in detail in the next chapter.

4-6 Truth Models

The initial truth model used for this design problem was simply the design model. To test the system more realistically, a nine-state truth model was developed with the additional states resulting from the addition of actuator dynamics. The actuators for the canards and stabilators were originally given as third order, but an

analysis of the dynamics of the actuators (shown in Appendix E) using the CAD package TOTAL [16], indicated no significant difference between the response using the third order actuators and a second order approximation. The resultant truth model used for controller evaluation is

$$\dot{\underline{x}}_t = A_t\underline{x}_t + B_t\underline{u}_t + \underline{w}_t \qquad (4\text{-}15)$$

where x_{t1} through x_{t4} are the original states of the system, x_{t5} and x_{t6} are actuator states involving canard deflection, x_{t7} and x_{t8} are actuator states for the stabilator, and x_{t9} is for the reverser vanes. The defining matrices of Equation (4-15) are:

$$A_t = \left[\begin{array}{cccc|cc|cc|c} A_{11} & A_{12} & A_{13} & A_{14} & B_{11} & 0 & B_{12} & 0 & B_{13} \\ A_{21} & A_{22} & A_{23} & A_{24} & B_{21} & 0 & B_{22} & 0 & B_{23} \\ A_{31} & A_{32} & A_{33} & A_{34} & B_{31} & 0 & B_{32} & 0 & B_{33} \\ A_{41} & A_{42} & A_{43} & A_{44} & B_{41} & 0 & B_{42} & 0 & B_{43} \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -8356 & -303 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -8356 & -303 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -89 \end{array}\right] \qquad (4\text{-}16)$$

$$B_t = \left[\begin{array}{ccc} & \underline{0} & \\ \hline 8356 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 8356 & 0 \\ 0 & 0 & 89 \end{array}\right] \qquad (4\text{-}17)$$

with $\underline{Q}$ a 5 x 3 zero matrix. As with the design model, initially the noise strength, Q_t, associated with $\underline{w}_t$ was set to zero. In the A_t matrix, the values of A_{xx} and B_{xx} are the respective matrix elements of the A and B matrices for the design condition, or the appropriate flight condition in the case of robustness tests.

4-7 Design Methodology

The design of the flight controller was accomplished in several steps using the CGTPIV software that was developed by Floyd [11] and Moseley [25] and modified by Miller [24]. The first step involved developing a CGT/PI full-state feedback controller without implicit model following, but with explicit model following, and evaluating the controller with respect to an environment as produced by the design model. The next step was to include an implicit model in the design process, but still only evaluate the resulting controller with respect to the design model. Next the controllers were tested against the truth model for the design condition, and finally uncertainty was added to the problem in the form of the dynamics noise in the system and imperfect noise-corrupted measurements to replace the artificial access to all states. The latter aspect led to the development of the Kalman filter. The outputs from CGTPIV, namely a CGT/PI controller and a Kalman filter, were then cascaded, and the resulting controller was evaluated using PFEVAL, a

covariance analysis package developed by Moseley [25] that assumes a linear truth model, and using ODEF15, a modification of the Monte Carlo analysis tool ODEACT developed by Miller [24] that permits nonlinearities (as particularly saturations) in the truth model definition. The actual designs and analysis will be covered in detail in the next chapter.

4-8 Summary

This chapter introduced the models used to develop the flight controller for the STOL F-15. The design, explicit command, implicit command, and truth models were discussed. The methodology employed in the design process was outlined as a preview for the next chapter.

V. ANALYSIS OF DESIGN RESULTS

5-1 Introduction

This chapter presents an analysis of the results of this study. The first section deals with the software used to evaluate the designed controllers. Then, the design path is presented in chronological order, with an analysis of the resulting controller. Finally, attempts at robustness enhancement, using the Doyle and Stein LTR technique, are discussed.

5-2 Performance Analysis Tools

The controllers designed in this study were evaluated using two software analysis tools developed in previous AFIT theses (and briefly introduced in Chapter 4). The first of these is PFEVAL, an interactive software tool used to conduct performance evaluations of linear sampled-data controllers against linearized models of realistic environments. The second is a modification of ODEACT, an interactive program that was originally written to account for actuator nonlinearities in the simulation of the real-world environment and to allow the addition of anti-windup compensation.

5-2.1 PFEVAL

PFEVAL was written by Moseley [25] to allow for the evaluation of stochastic controllers that was not available with CGTPIF [11], namely the evaluation of a filter/

controller cascade as tested against a higher order "truth model" (Fig. 3-2). To accomplish this, he ammended CGTPIF so that data required for performance evaluation would be saved in the output file. This output file could then be used as a data file in PFEVAL. PFEVAL, with the initial covariances for the truth states and control inputs supplied by the user, performs evaluations of the controller with and without the filter embedded. The output of PFEVAL is a time history, for fifty sample periods, of the standard deviations of all truth states and control inputs for the controller, with and without the filter, and a list of final estimation error covariances for both.

5-2.2 ODEF15

ODEF15 is a modification of the nonlinear simulation package ODEACT, originally written by Miller [24]. The initial version of this program was used to extend the analysis of a controller, designed via CGTPIF, to include nonlinear effects generated by the inclusion of actuator saturations, specifically for the AFTI F-16. For the purpose of this research, the program was modified, and renamed ODEF15, to allow for the actuator saturation nonlinearities of the STOL F-15, the possible inclusion of a Kalman filter with Monte Carlo analysis, and the output of CALCOMP plots. A listing of the ODEF15 code and a sample run of the program are included in Appendix A.

The gain matrices for controllers and filters designed with CGTPIF are entered interactively into ODEF15 along with the continuous-time system matrices of Equations (4-3), (4-4), and (4-6), the discrete-time E_d and B_d, which are outputs of CGTPIF, and the strength of the measurement noises, as given in Equation (4-8). The program is set up to perform interactive simulation runs with or without the inclusion of the Kalman filter, rate and position limits, actuator dynamics, and anti-windup compensation.

5-3 Controller Design

The controller design, using CGTPIF, is done in a building block fashion. Initially, the state-space matrices, A, B, and C, of Equations (4-3), (4-4), and (4-8), are entered into CGTPIF. As the design system is unstable, an open-loop CGT design is not feasible, so a PI controller is first designed and then the closed-loop CGT. This initial design is accomplished without implicit model following and is not evaluated against a truth model. The initially chosen values for the weighting matrices were based upon the physical characteristics of the plant. In the case of the output deviation weights, since the main concern is for the control of flight path angle, it was initially weighted with a coefficient of three as compared to one for the velocity and pitch rates. As the canard/flap/aileron combination and the stabilator are limited to approximately the same amount of deflection from

the equilibrium and the reverser vanes can move almost twice as much, the control magnitude weights are initialized at two, two, and one respectively. The control rate weights are set initially at eight to four to one for the canard/ flap/aileron combination, stabilator, and engine reverser vanes respectively, as this is the ratio of the maximum deflection rates for these surfaces. Another choice for the weightings [20] would be to use the square of the differences in "concern," position limits, or rate limits, but as these are only the starting points for a control weight tuning process, the weightings listed in this paragraph are used. This design yields:

$$K_x = \begin{bmatrix} .0081849 & 5.182 & -22.8 & .092615 \\ .044529 & -13.34 & 25.96 & -.2100 \\ -1.977 & 2.927 & 3.780 & .034236 \end{bmatrix} \quad (5\text{-}1)$$

$$K_z = \begin{bmatrix} .014268 & 1.038 & .4107 \\ -.0034988 & -1.628 & -.5825 \\ -.5333 & -.089242 & -.063284 \end{bmatrix} \quad (5\text{-}2)$$

in the controller of Equation (2-68). Next, a design of the CGT is pursued with the explicit model as shown in Section 4-4. This design yields:

$$K_{xm} = \begin{bmatrix} .0083856 & -25.19 & -.2480 & -.090792 \\ .044515 & 22.16 & .5140 & .099918 \\ -1.980 & 4.165 & -1.385 & .010521 \end{bmatrix} \quad (5\text{-}3)$$

$$K_{xu} = \begin{bmatrix} 2.069 \\ -2.427 \\ -.2046 \end{bmatrix} \qquad (5\text{-}4)$$

in the controller of Equation (2-68), where:

$$K_{xm} = K_x A_{11} + A_{21} \qquad (5\text{-}5a)$$

$$K_{xu} = K_x A_{12} + A_{22} \qquad (5\text{-}5b)$$

Printouts of the time response of the output variables and the control deflections with the initial CGT/PI full-state feedback controller are shown in Figures 5-1 and 5-2. It can be seen that the flight path angle is slow, for a fighter type aircraft, to attain the commanded value of a five degree downward flight path. The controller is otherwise well behaved as it neither violates nor approaches any physical constraints of the system. Further iterations in this step of the design process led to the use of the following weights for the cost function of Equation (2-5): for output deviations, 1:1:25 for velocity, pitch rate, and flight path angle, respectively; for command input magnitude deviations, 10:3:1 for the canard/flap/aileron combination, stabilator, and reverser vanes, respectively; and for command input rate deviations for the canard/flap/aileron, stabilator, and reverser vanes, 60:30:1, respectively. These weights are to be used in the next step in this process. Figures 5-3 and 5-4 show a time history for the output variables and the control deflections for this controller. This controller exhibits a much faster settling time with approximately the same

INITIAL CONTROLLER

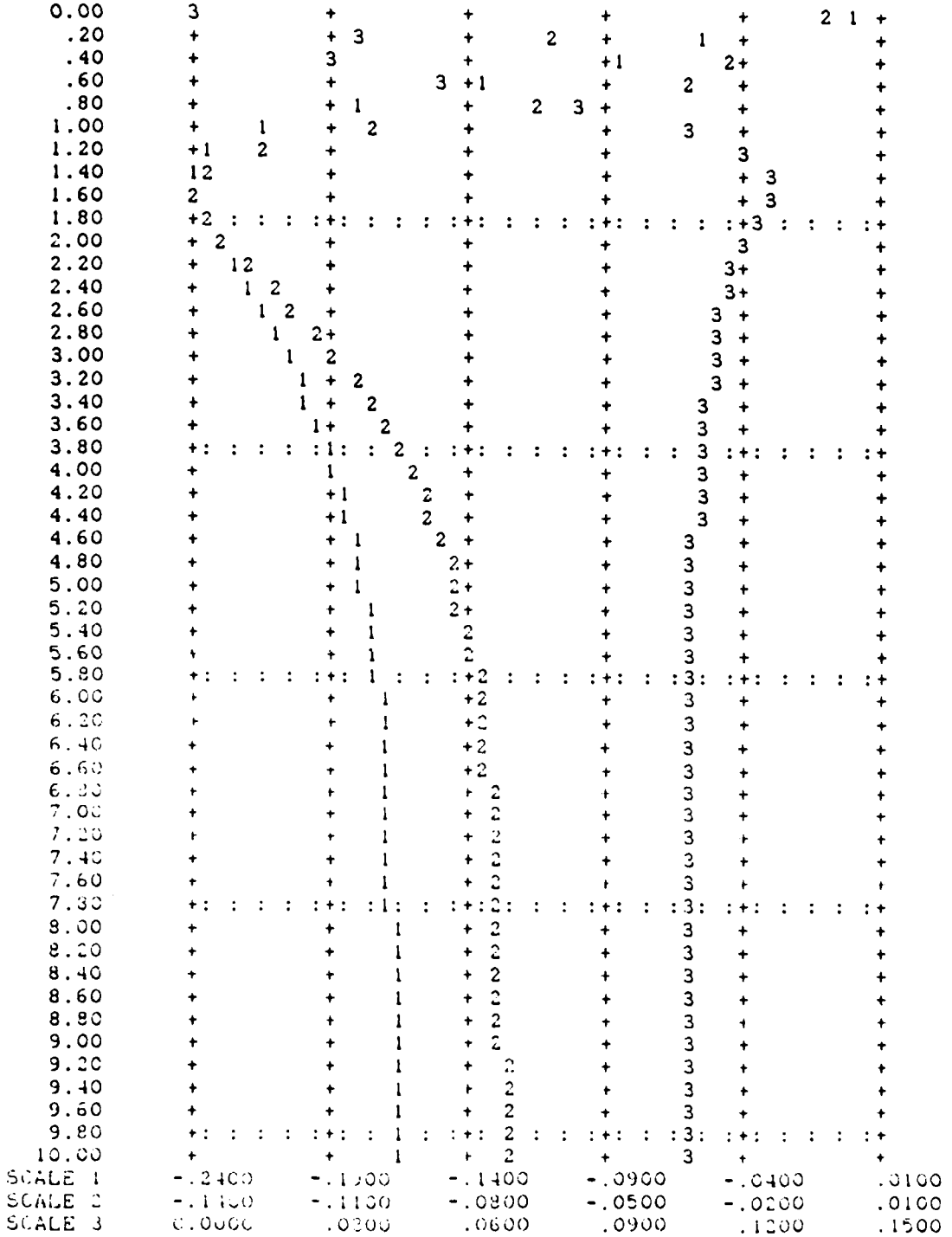


Figure 5-1. Output Variables for Initial Controller (1) Velocity (fps) (2) Pitch Rate (rad/sec) (3) Flight Path Angle (rad)

1 INITIAL CONTROLLER

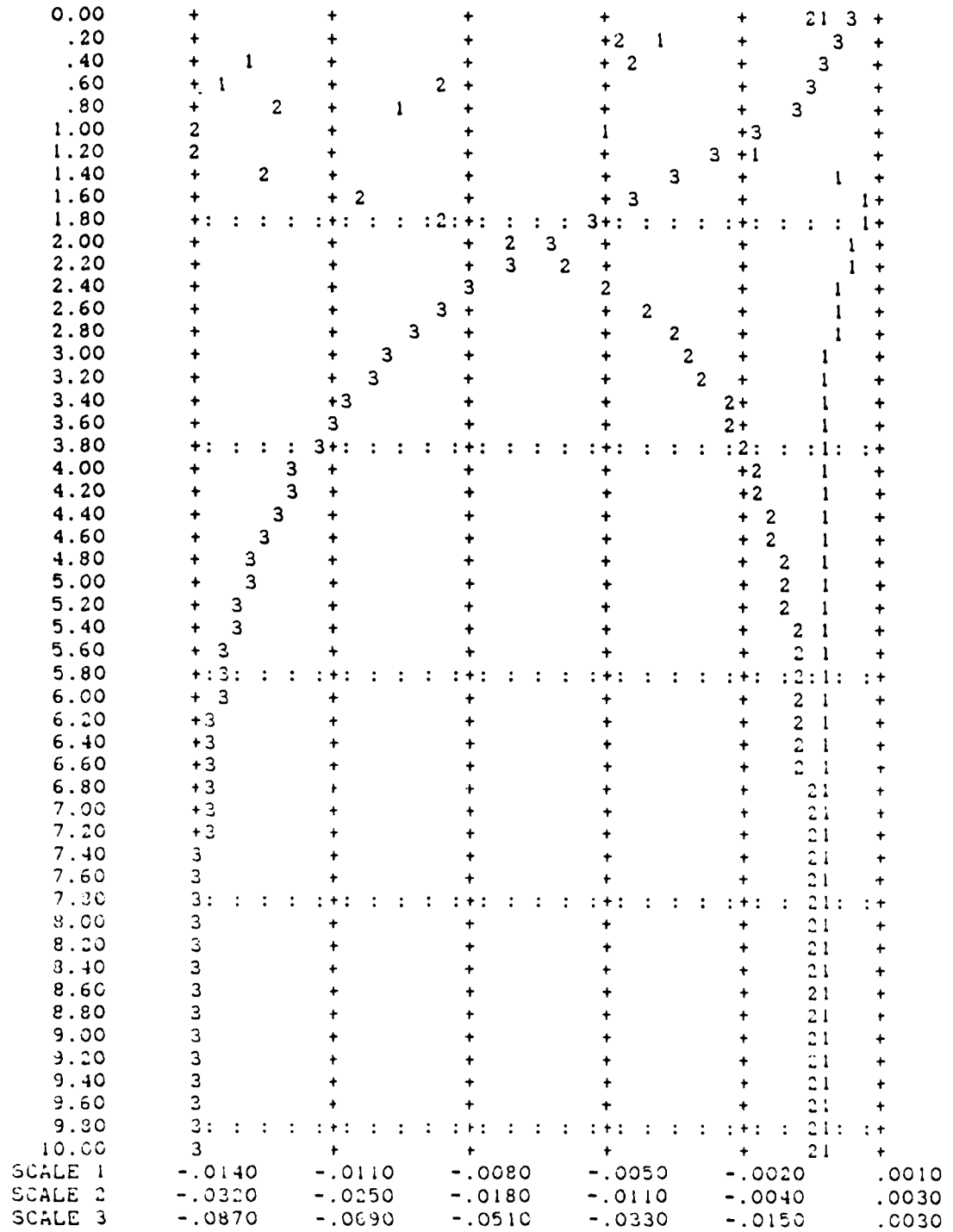


Figure 5-2. Control Deflections for Initial Controller (1) Canard (2) Stabilator (3) Reverser Vanes (radians)

CONTROLLER WITHOUT IMPLICIT MODEL FOLLOWING

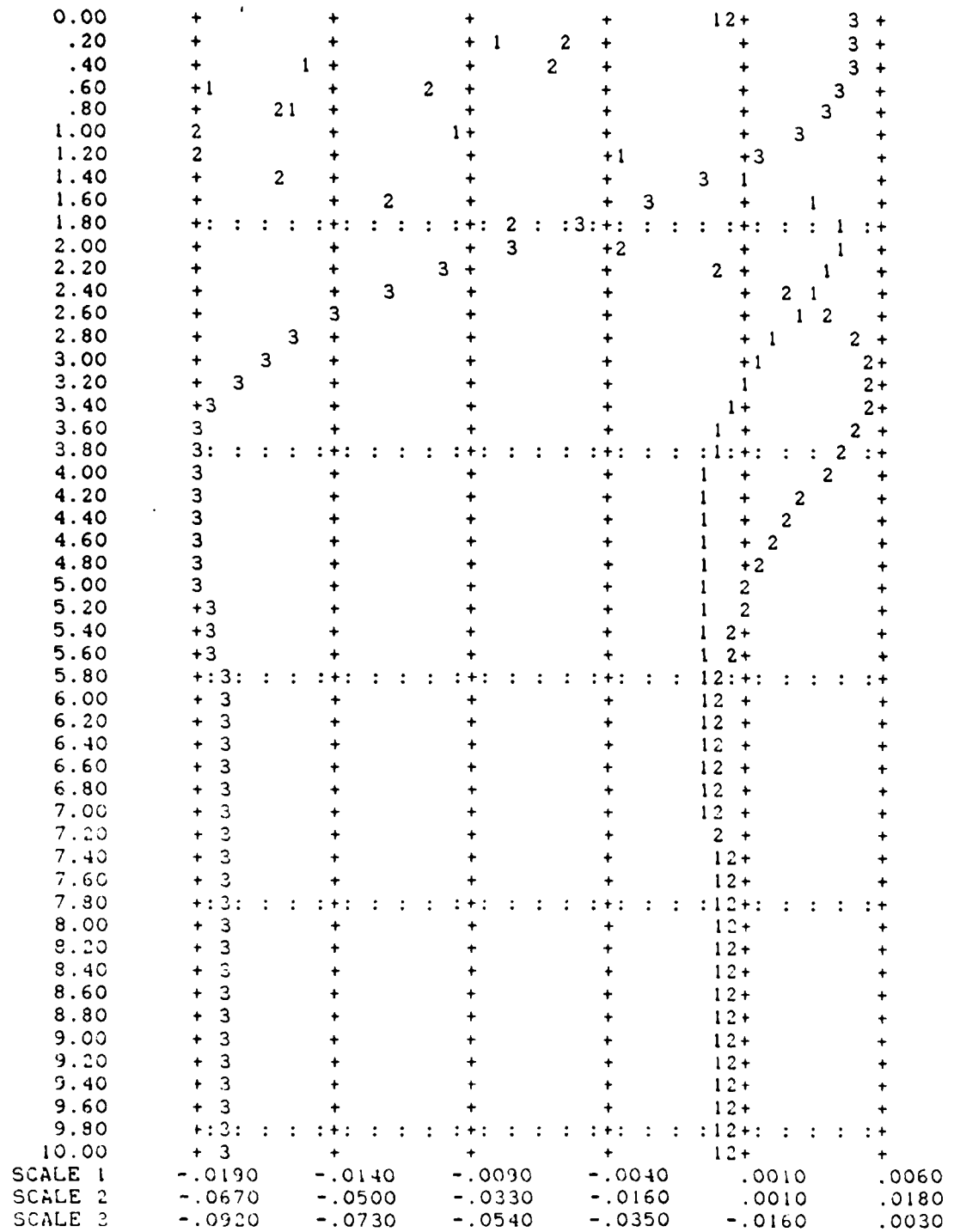


Figure 5-3. Output Variables for Controller Without Implicit Model Following

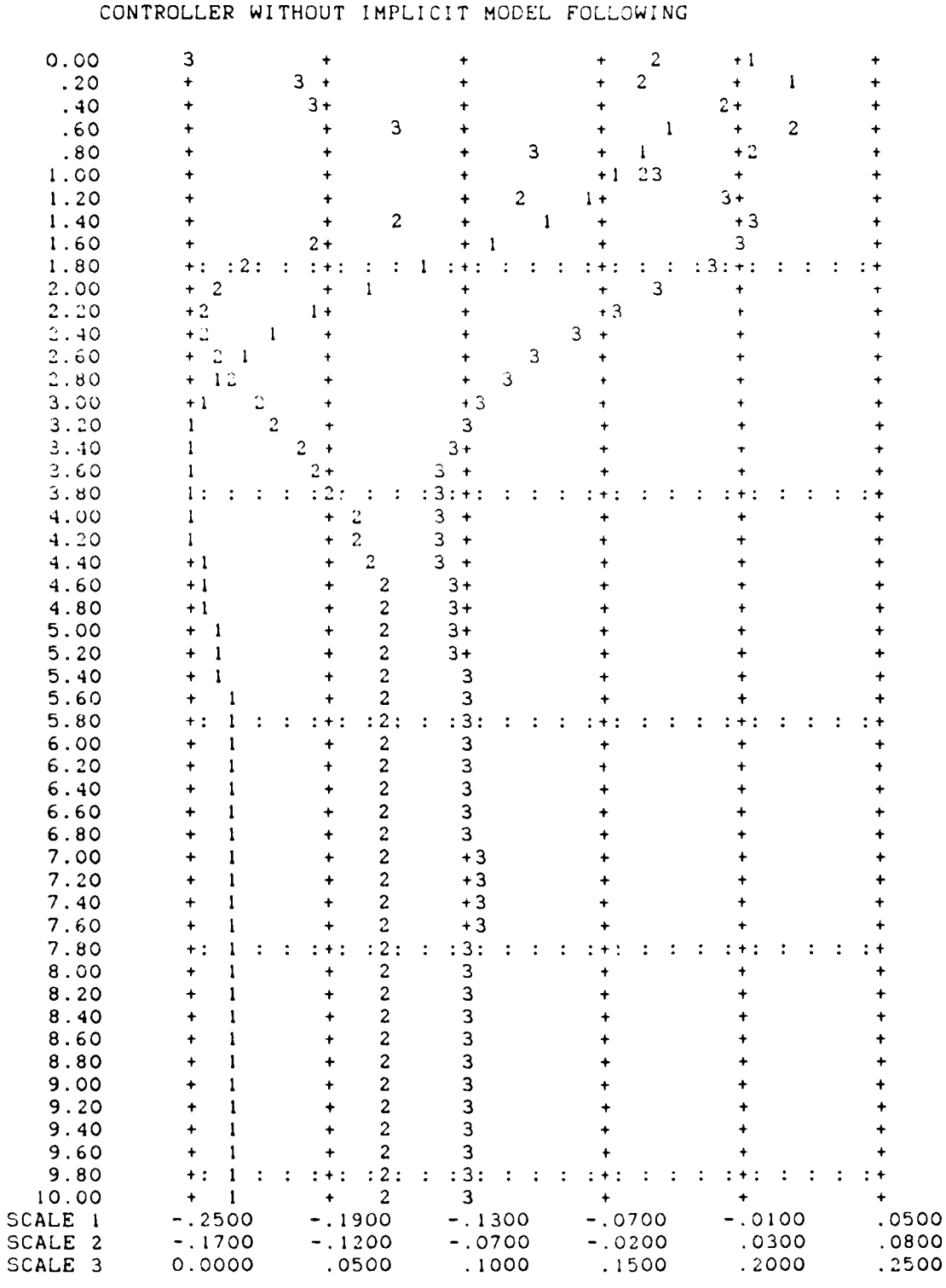


Figure 5-4. Control Deflections for Controller Without Implicit Model Following

amount of control energy as the initial controller. This performance is somewhat misleading as the controller is being evaluated against a four-state "truth model" and significant degradation in performance can be expected when a higher dimension truth model is used.

This next step in the iteration is to include an implicit model in the PI controller. The model used is the one in Section 4-5 with the diagonal elements all set to negative five and the weights on the derivatives of the outputs and the control pseudo-rates set to one. The controller was developed with the same explicit model and weightings as above with the resulting gain matrices:

$$K_x = \begin{bmatrix} .0093445 & 4.304 & -22.82 & .080089 \\ .020574 & -11.21 & 30.32 & -.1854 \\ -3.571 & 1.608 & -2.932 & .025107 \end{bmatrix} \quad (5\text{-}6)$$

$$K_z = \begin{bmatrix} -.0043609 & .9997 & .5725 \\ .0038191 & -1.701 & -.7030 \\ -.4240 & .1885 & .0066257 \end{bmatrix} \quad (5\text{-}7)$$

$$K_{xm} = \begin{bmatrix} .0095451 & -26.11 & -.2610 & -.091421 \\ .020566 & 30.22 & .6369 & .1197 \\ -3.574 & -6.221 & -1.545 & -.018581 \end{bmatrix} \quad (5\text{-}8)$$

$$K_{xu} = \begin{bmatrix} 2.072 \\ -2.825 \\ .4060 \end{bmatrix} \quad (5\text{-}9)$$

Plots of the time response of the system outputs and control inputs, now evaluated against the nine-state truth model of Equation (4-15), are shown for this controller in Figures 5-5 and 5-6. The addition of the truth model to the evaluation introduces the effects of actuator dynamics and will, therefore, have a negative impact on system response compared to an evaluation against a "truth model" equated to the design model. While this controller shows an improvement in settling time, both position and rate limits for the canard/flap/aileron combination and rate limits for the stabilator are violated. To alleviate these problems, the implicit model is changed to reflect the desire to slow down the inputs to these control surfaces, which requires a trade-off in performance. The resulting controller has the same explicit weighting matrices for the performance index of Equation (2-72a) as the initial controller, with the final implicit weights on output derivatives set at 1:1:2 for the velocity, pitch rate, and flight path angle, respectively, and the final implicit control pseudo-rate weights at 2:2:1 for the canard/flap/aileron combination, stabilator, and reverser vanes, respectively. The implicit command model is:

$$A_m = \begin{bmatrix} -.05 & 0 & 0 \\ 0 & -.05 & 0 \\ 0 & 0 & -.5 \end{bmatrix} \qquad (5\text{-}10)$$

The difference in the last element of A_m is from the desire to obtain a faster response in flight path angle than in

CONTROLLER WITH IMPLICIT MODEL FOLLOWING

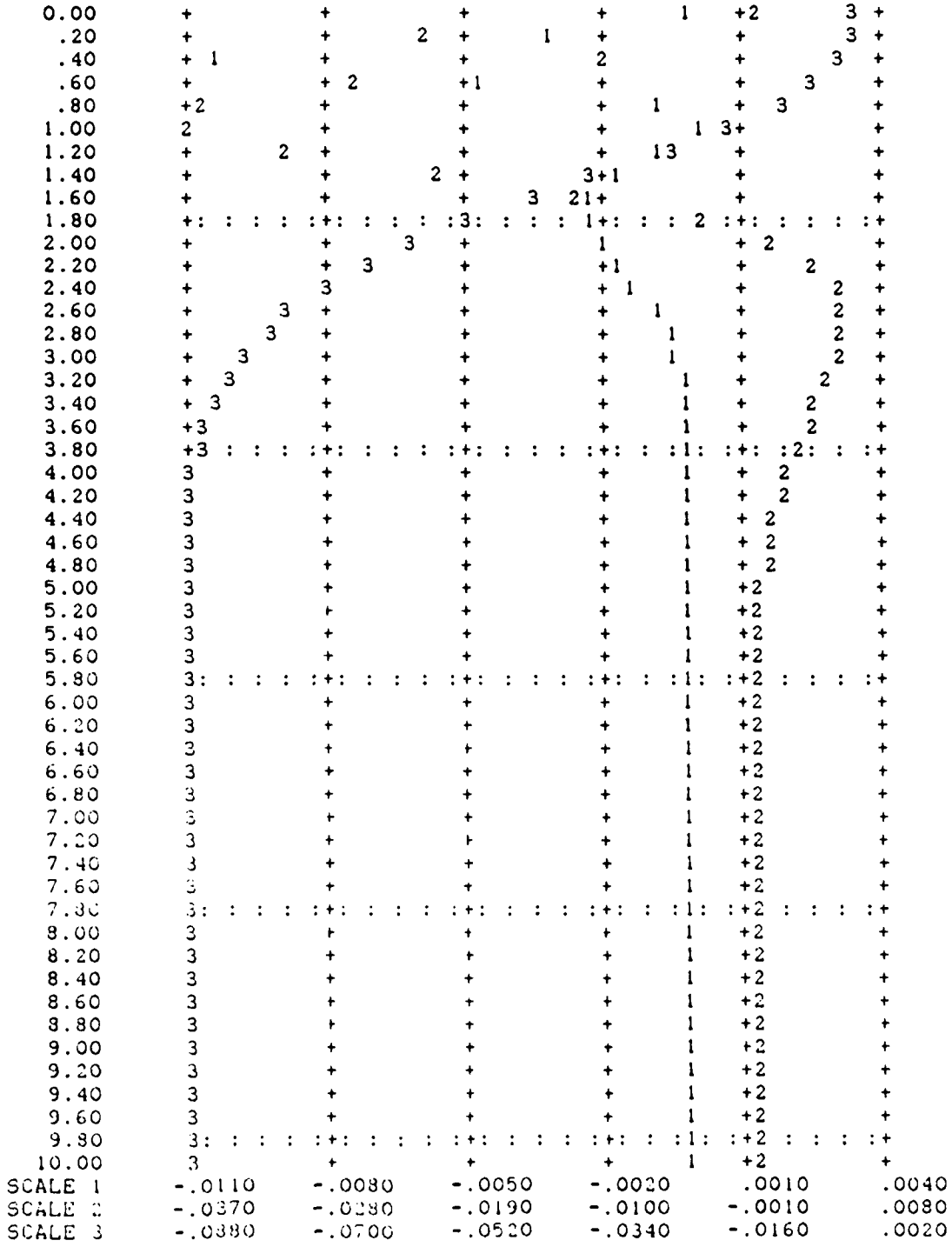


Figure 5-5. Output Variables for Controller With Implicit Model Following

CONTROLLER WITH IMPLICIT MODEL FOLLOWING

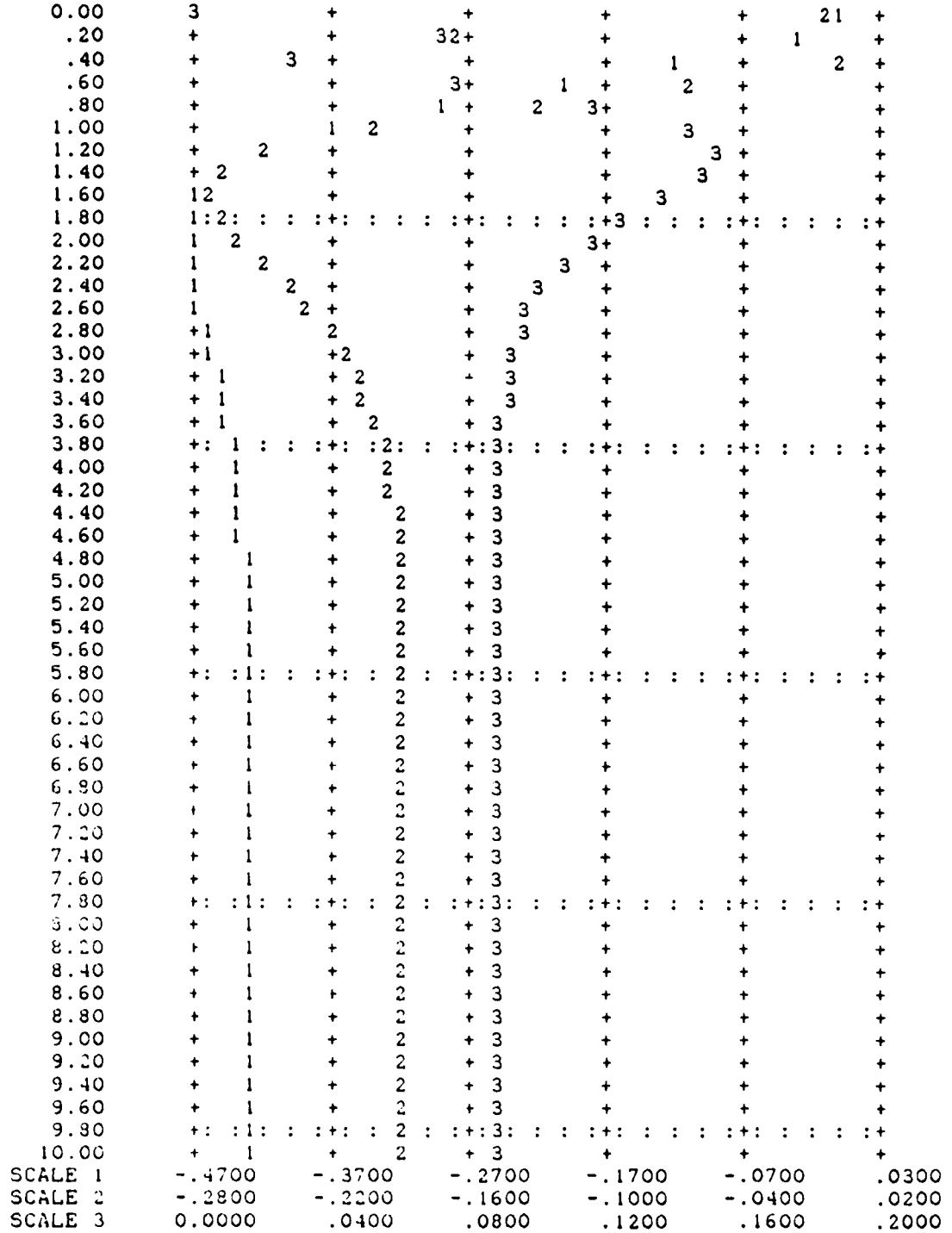


Figure 5-6. Control Deflections for Controller With Implicit Model Following

velocity or pitch rate. These values yield a controller with gain matrices of:

$$K_x = \begin{bmatrix} .024718 & 4.532 & -12.55 & .072426 \\ .027933 & -10.94 & 27.59 & -.1744 \\ -3.404 & 1.466 & -1.328 & .020731 \end{bmatrix} \quad (5\text{-}11)$$

$$K_z = \begin{bmatrix} -.0031405 & .5861 & .3119 \\ .011646 & -1.405 & -.6242 \\ -.083988 & .089413 & .034607 \end{bmatrix} \quad (5\text{-}12)$$

$$K_{xm} = \begin{bmatrix} .024932 & -12.04 & -.038514 & -.04804 \\ .027922 & 26.81 & .5866 & .1083 \\ -3.407 & -4.205 & -1.513 & -.011917 \end{bmatrix} \quad (5\text{-}13)$$

$$K_{xu} = \begin{bmatrix} 1.140 \\ -2.578 \\ .2606 \end{bmatrix} \quad (5\text{-}14)$$

Time response plots for this controller, given a five degree nose-down flight path angle command, are shown in Figures 5-7 and 5-8. This controller provides a reasonable settling time with a very mild overshoot and violates none of the physical constraints imposed upon the system. As this design satisfies the desire for performance without violating, or approaching, the limits of the system, it is used for the analysis to follow.

It is at this juncture that the assumption of full-state feedback is replaced with the more realistic

FINAL CONTROLLER

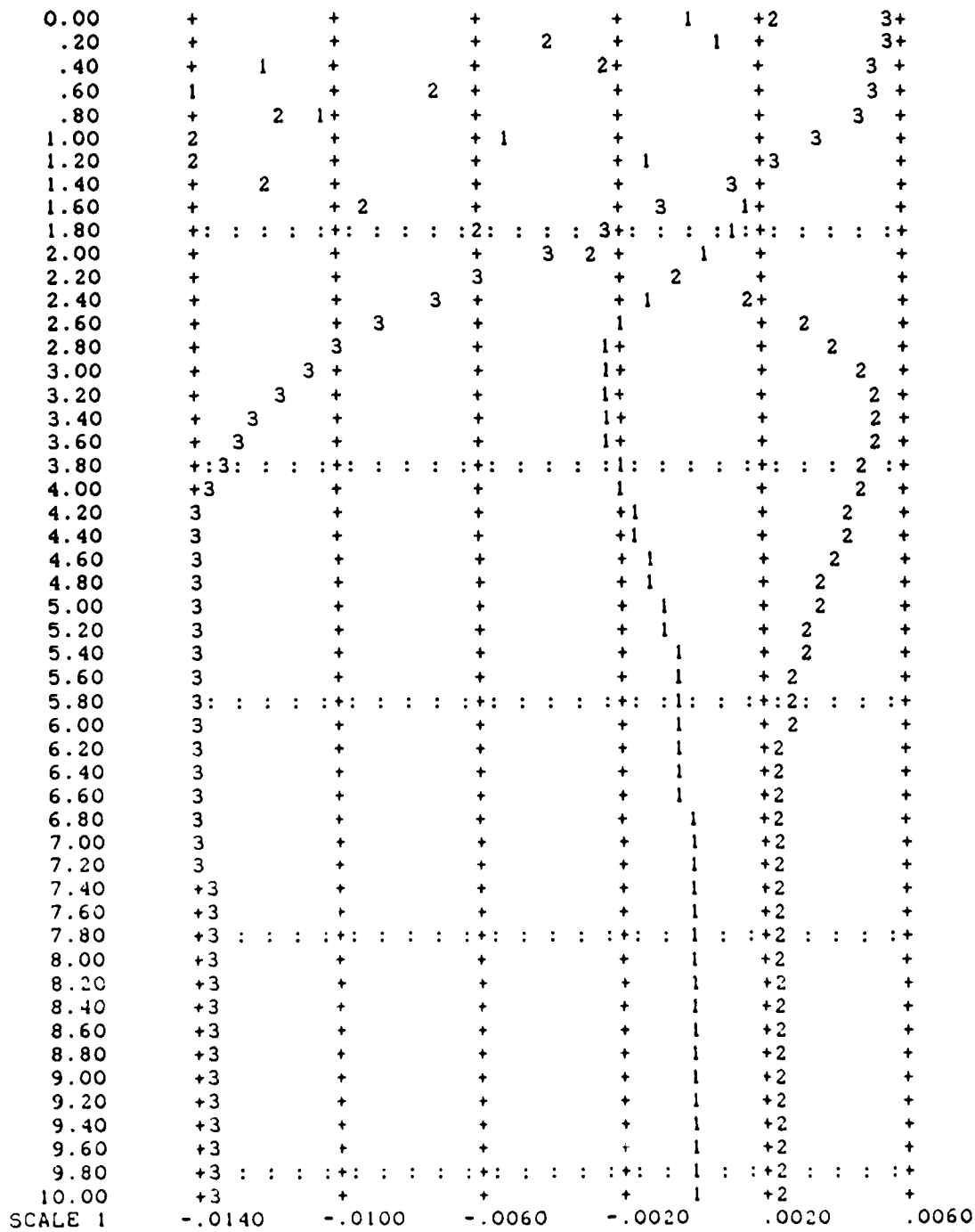


Figure 5-7. Output Variables for Final Controller

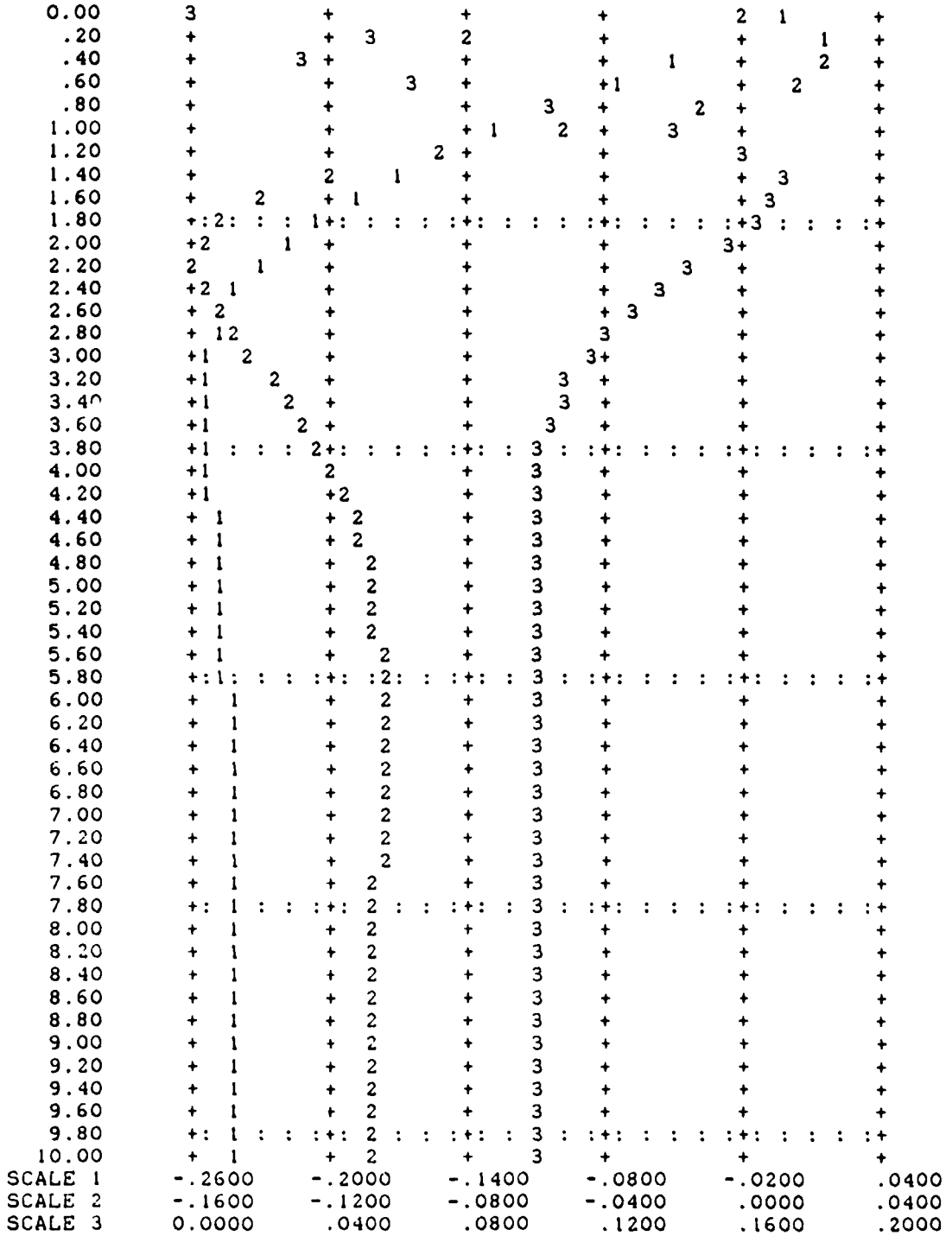


Figure 5-8. Control Deflections for Final Controller

assumption of incomplete/imperfect measurements and the addition of a Kalman filter. For this purpose the measurement matrix H and the sensor noise matrix R, shown in Section 4-3, are included in the design model (and also in the truth model). In addition, the G and G_t matrices are included in the design and truth models, respectively. In both cases they are identity matrices of the appropriate dimension. Initially, noises are associated with the angle of attack and pitch rate and their strengths are set to unity as a first guess, and, after numerous iterations between CGTPIF and ODEF15, to attain the desired performance both took on the value of .0001. With these additions to the design and truth models, it is possible to design and evaluate a Kalman filter using CGTPIF. After a number of iterations to attain good estimation precision and controller performance at "on-design" conditions, the CGT/PI/KF controller to be used as a baseline in the filter-in-the-loop robustness enhancement studies was derived. The controller of Equations (5-10) through (5-14), with final Kalman filter gains of:

$$K_F = \begin{bmatrix} .6153 & -.1122 & .0019542 \\ -.2875 & .1958 & -.0072972 \\ -.02827 & .0221 & -.2300 \\ -.015051 & .0023899 & -.21043 \end{bmatrix} \qquad (5\text{-}15)$$

is to be evaluated with PFEVAL and ODEF15 against a number of varying real-world conditions.

5-4 Performance Analysis

The controller developed in the previous section is analyzed in detail in this section. First, a linear covariance analysis is accomplished using PFEVAL. Then, the nonlinear simulation program, ODEF15, is used to test the control laws against a more realistic environment including actuator saturations. Finally, LTR tuning is used in an attempt to recover some of the robustness lost in going from full-state feedback to the controller with a Kalman filter in the loop.

The results of the covariance analysis at the final simulation time are presented in Table 5-1 and show that the inclusion of the filter, with state estimates updated with the incoming measurements, improves the performance of the controller. This is not a totally unexpected result, as the truth model against which the controller is being evaluated includes actuator dynamics (see Section 4-6), and therefore involves effects not modelled in the design of the controller; if the truth model were the same as the design model, the full-state feedback design would naturally outperform the design model with the filter in the loop.

Table 5-1

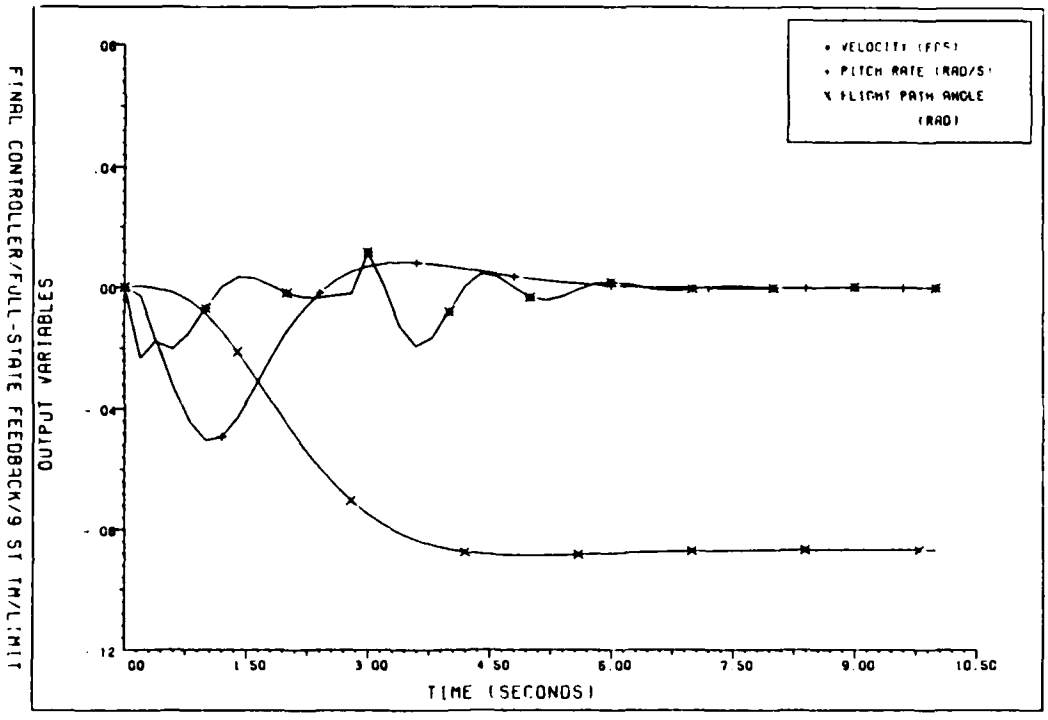
Final Standard Deviations of CGT/PI/KF Controller Design

TRUTH STATE	STANDARD DEVIATIONS (σ IN RADIANS OR RADIANS/SEC)	
	WITH FILTER	WITHOUT FILTER
U	4.9617264×10^{-3}	4.165973×10^{-3}
q	1.2784801×10^{-2}	1.3747698×10^{-2}
α	1.1799363×10^{-2}	1.2151774×10^{-2}
θ	7.0565759×10^{-3}	7.5290352×10^{-3}
δ_{cfa}	2.1194026×10^{-2}	2.8109224×10^{-2}
$\dot{\delta}_{cfa}$	3.3509912×10^{-1}	4.4106762×10^{-1}
δ_s	5.1844835×10^{-2}	6.8388985×10^{-2}
$\dot{\delta}_s$	7.6988819×10^{-1}	1.0284194
δ_{rv}	3.1738364×10^{-2}	3.3501037×10^{-2}
δ_{cfaI}	2.5876006×10^{-2}	3.4182074×10^{-2}
δ_{sI}	6.1941621×10^{-2}	8.1942491×10^{-2}
δ_{rvI}	3.2065578×10^{-2}	3.3686558×10^{-2}

5-4.1 Nonlinear Performance Analysis

The robustness evaluation of the controller using ODEF15 is done in four segments. In the first section, the controller is tested using the design flight condition with actuator dynamics, i.e., a nine-state truth model, and rate and position limits considered. As the controller is designed for a five-degree nose- down flight path angle,

this is the first condition tested. The full-state feedback response and the response with the inclusion of a Kalman filter based on $\hat{\underline{x}}(t_i^+)$ are shown in Figures 5-9 and 5-10 respectively. These responses for the full-state feedback case are similar to those obtained when the full-state feedback controller was evaluated against the nine-state truth model in CGTPIF and, as the controller is designed to avoid the rate and position limits, this is to be expected. As the measurements of the outputs are very accurate, the results for $\hat{\underline{x}}(t_i^+)$ are expected to be close to that of full-state feedback, and this in fact occurs, as is seen in a comparison of the plots for the respective outputs. It should be noted that the differences in velocity are actually very small when compared with the flight velocity of 200 feet per second and that the apparent divergence occurs well out of the time period of interest, approximately six seconds. There is also significantly more control energy used with the Kalman filter than with the full-state feedback case. This is due to the estimation of state variables, the inclusion of actuator dynamics in the truth model, and the resulting lag in control application. The results for a controller based on $\hat{\underline{x}}(t_i^-)$ are not shown here, nor will they be shown in this research. This is because, in order to stabilize the controller, a value for Q was so small as to make the problem physically meaningless. This may be interpreted as the filter having almost no confidence in the dynamics



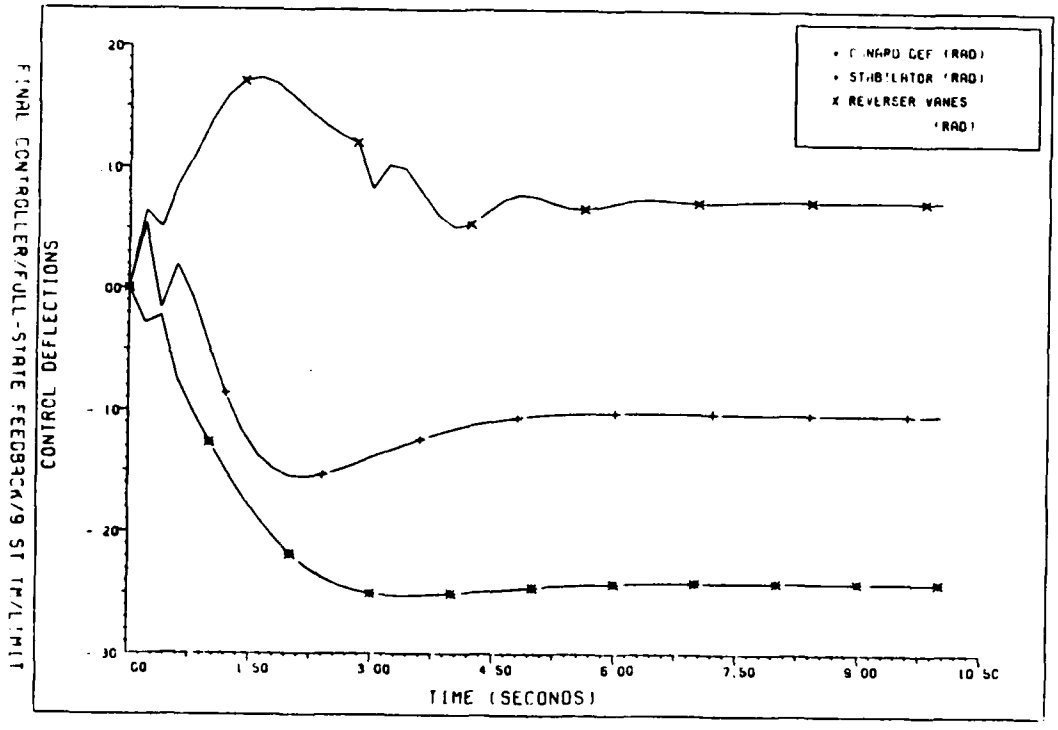


Figure 5-9. Full-State Feedback Controller

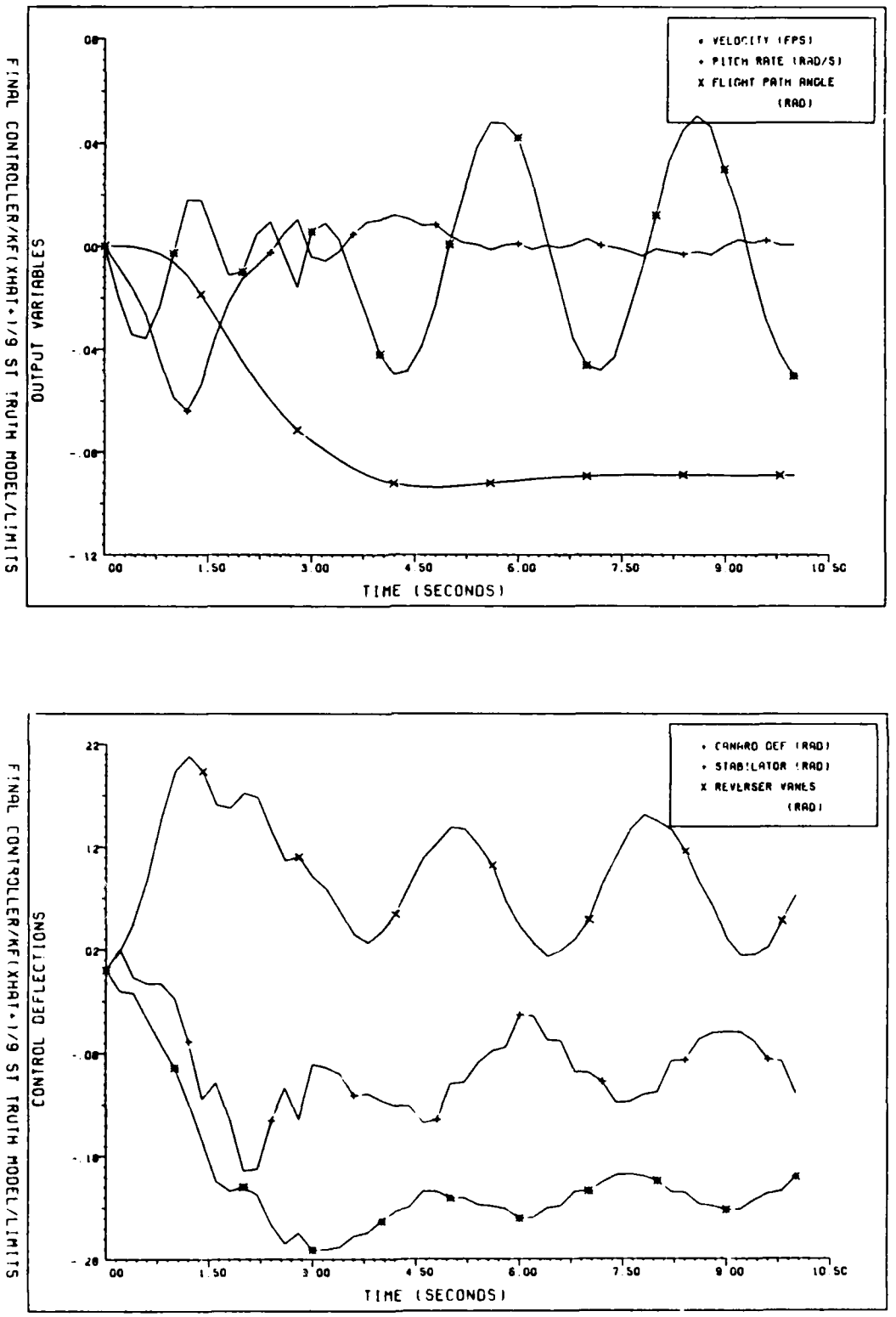


Figure 5-10. Controller with Kalman Filter in the Loop

model and putting extremely heavy weight on the incoming measurements. While the measurements for this study are very accurate, it is imprudent to ignore the embedded dynamics model completely. It is therefore decided to eliminate $\hat{\underline{x}}(t_i^-)$ from consideration in this problem, but with this in mind, a computationally efficient algorithm for the controller must be used. This algorithm must minimize the destabilizing delay between the incoming measurement, $\underline{z}(t_i)$, and the output of a control signal, $\underline{u}(t_i)$, and the total time for the computation must also be less than the sample period of the system.

The second block of evaluation was for inputs other than for what the controller was designed. For this and all subsequent evaluations, the controller with the filter-in- the-loop is considered. First, a shallower flight path angle (three degrees) was commanded. The results, shown in Figure 5-11, indicate that the controller is capable of complying with this command. Another analysis is done with the aircraft in equilibrium with a minus five degree flight path angle. In this case, a five degree pitch up command is given, and as seen in Figure 5-12, the controller has no difficulty with this task.

Next, system failures are simulated by first increasing the sensor noises and, secondly, simulating the loss of an actuator. For the section, sensor degradation is simulated by a two-order- of-magnitude increase in sensor

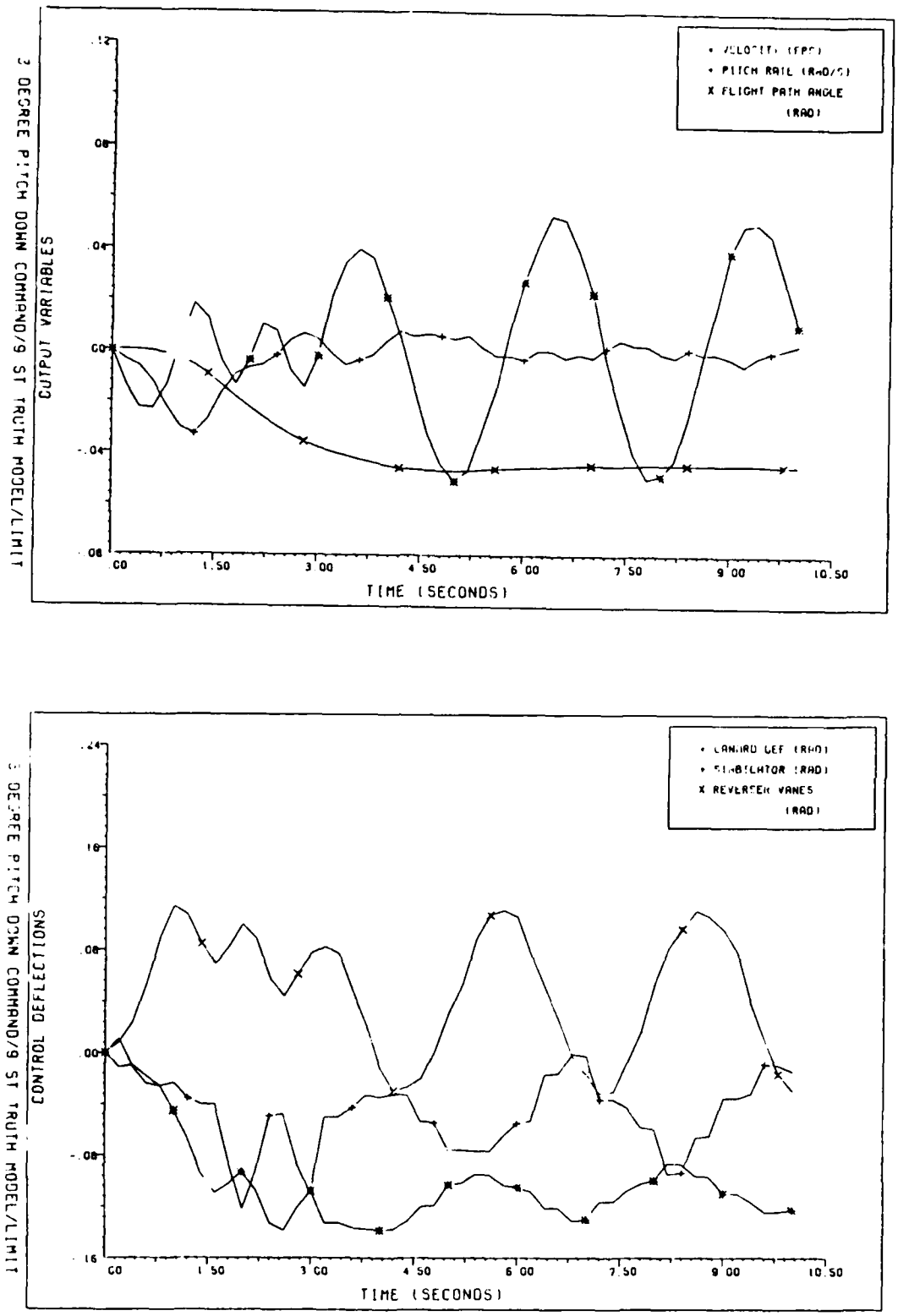
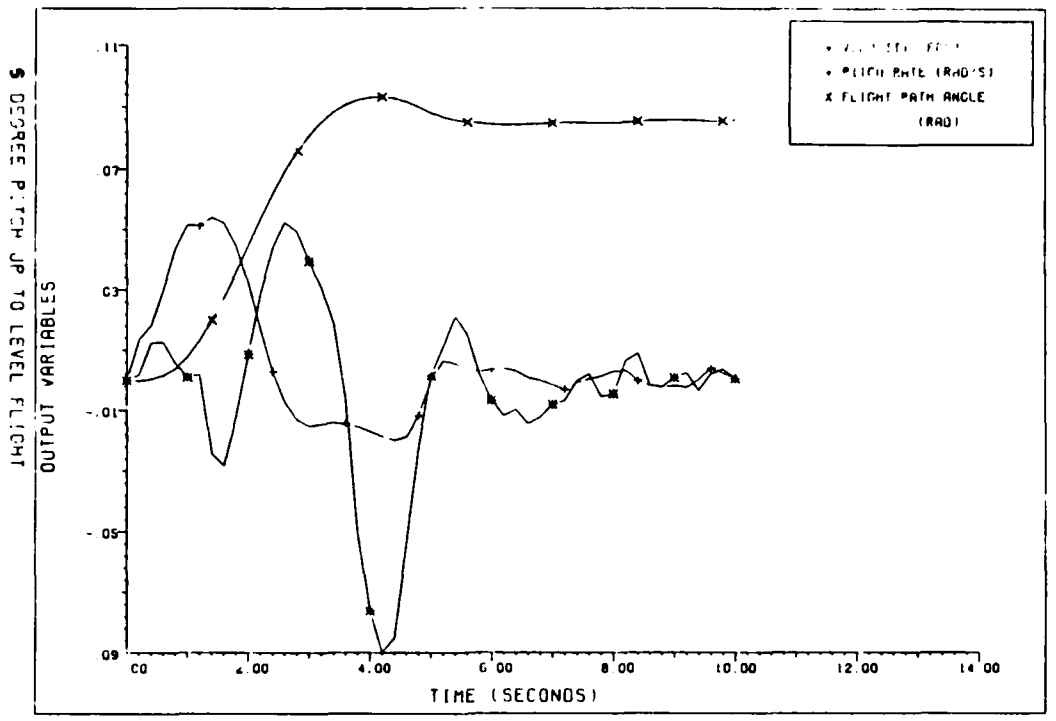


Figure 5-11. Three Degree Pitch Down Manuever



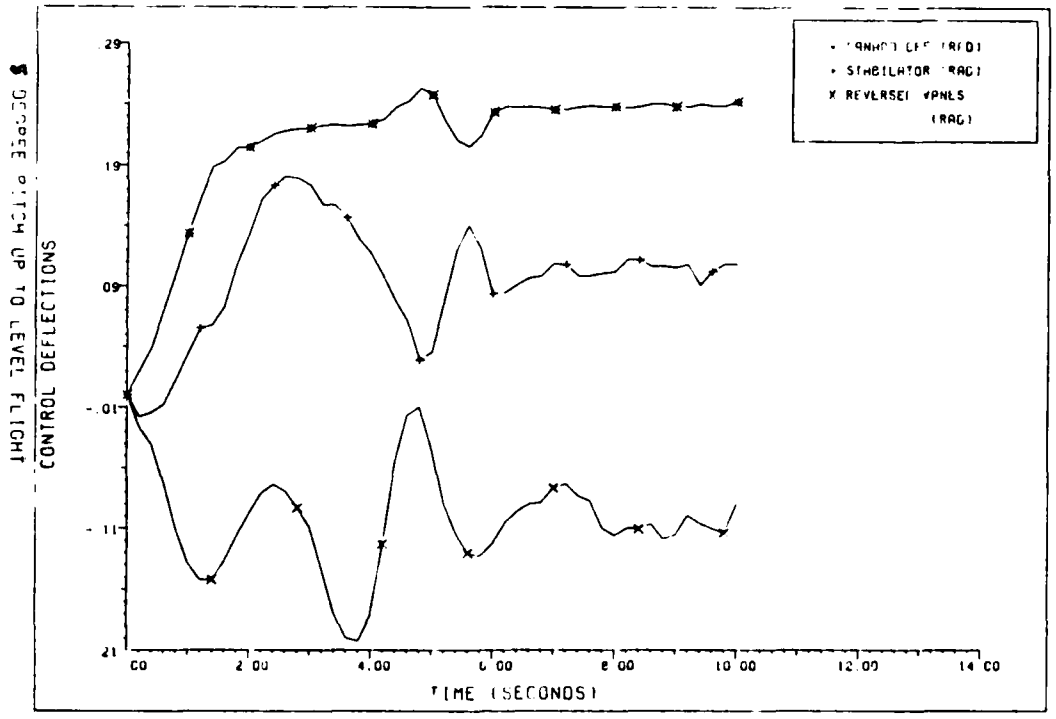


Figure 5-12. Five Degree Pitch Up to Level Flight

noise in the truth model without giving the filter an indication of the change. As is demonstrated in Figure 5-13, this noise increase has the effect of causing oscillations about the nominal values, but is not destabilizing at this magnitude. It exhibits characteristics similar to the filter with non-degraded sensor data, i.e. similar to the full-state feedback case in flight path angle, but more oscillatory in velocity and pitch rate and with more active control surfaces. The second system failure to be simulated is that of a canard failure. It is assumed that the canard is free-floating and contributes nothing to the moment generation required to pitch the aircraft. This is accomplished by zeroing out the command to the canard/flap/aileron combination in ODEF15. This slows the system down and also eliminates the overshoot seen in the case of a healthy system. Figures 5-14 and 5-15 depict the results of a five degree pitch down and a five degree pitch up to level-off, respectively, which can be compared to Figures 5-10 and 5-12. There is very little degradation in the flight path angle for either case, just slightly slower as was indicated above, and the other outputs and control deflections become more oscillatory in the face of the failure. This indicates a great deal of robustness.

In the final section of the performance evaluation, the controller is tested against off-design flight conditions. This has, to some degree, already been done,

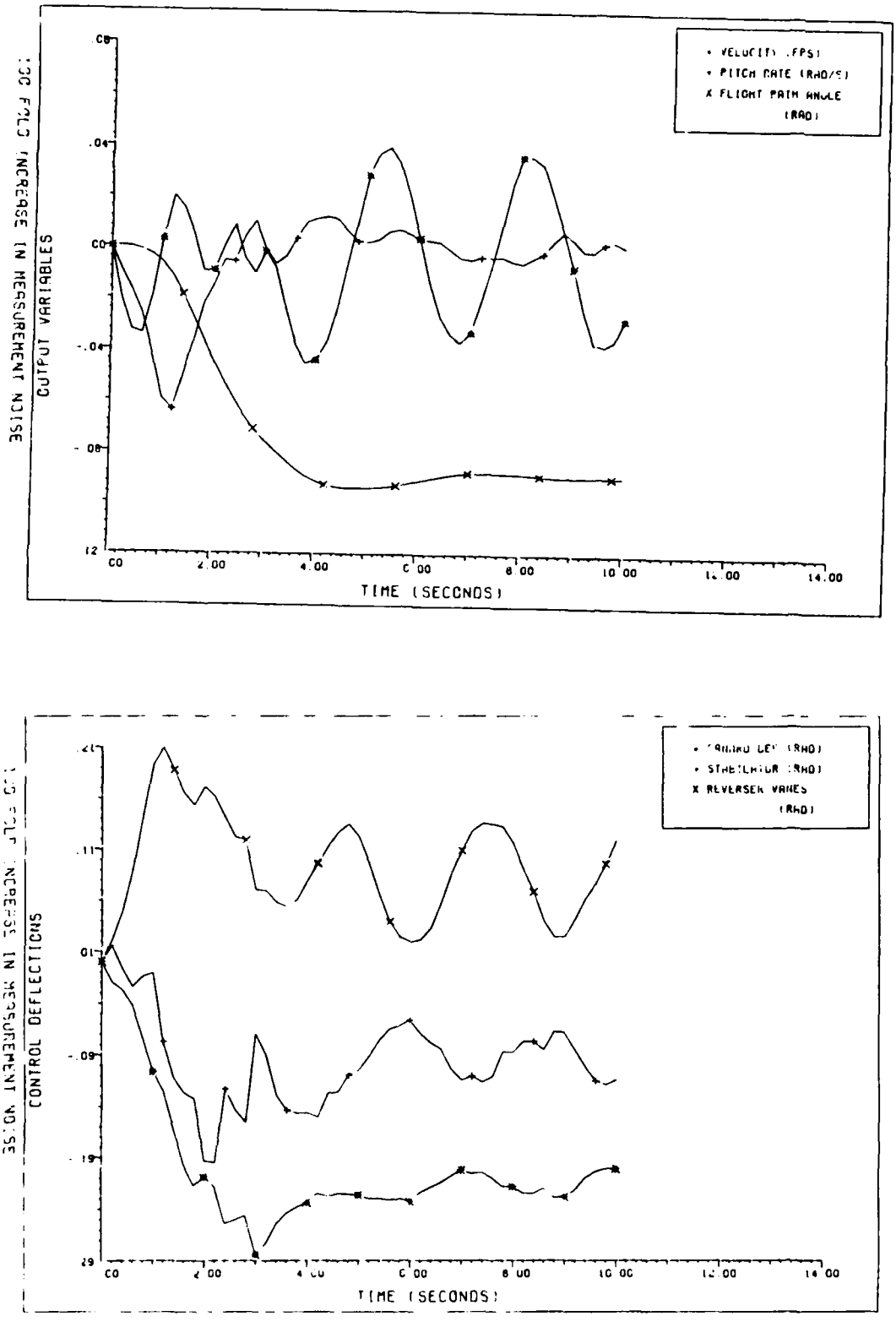
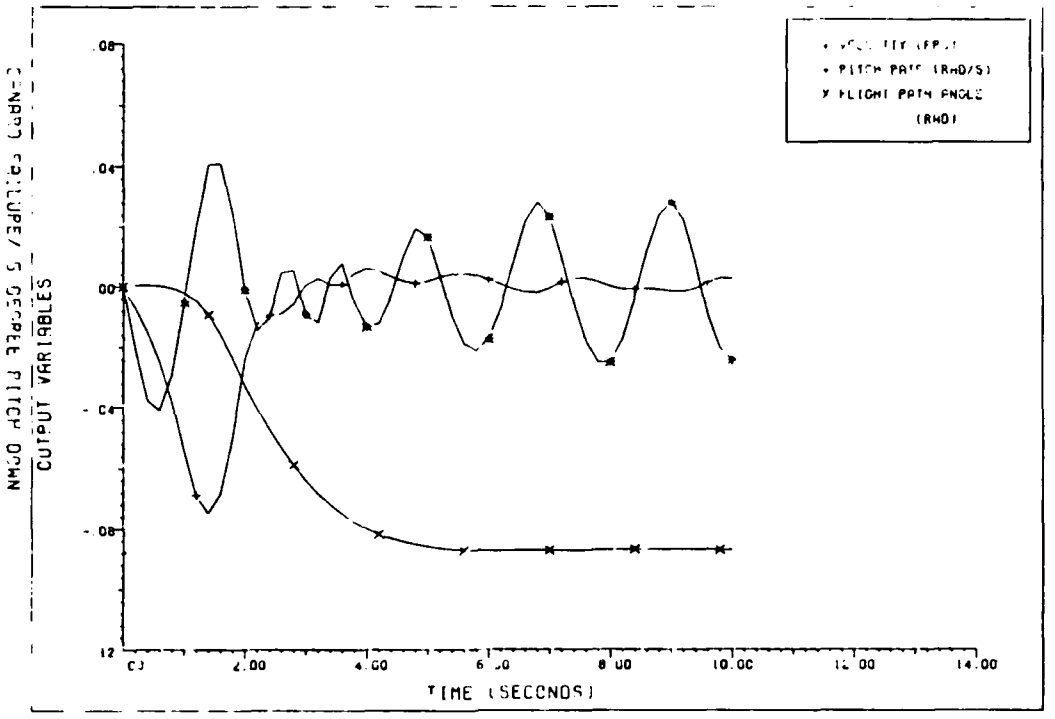


Figure 5-13. Effect of Degraded Sensors



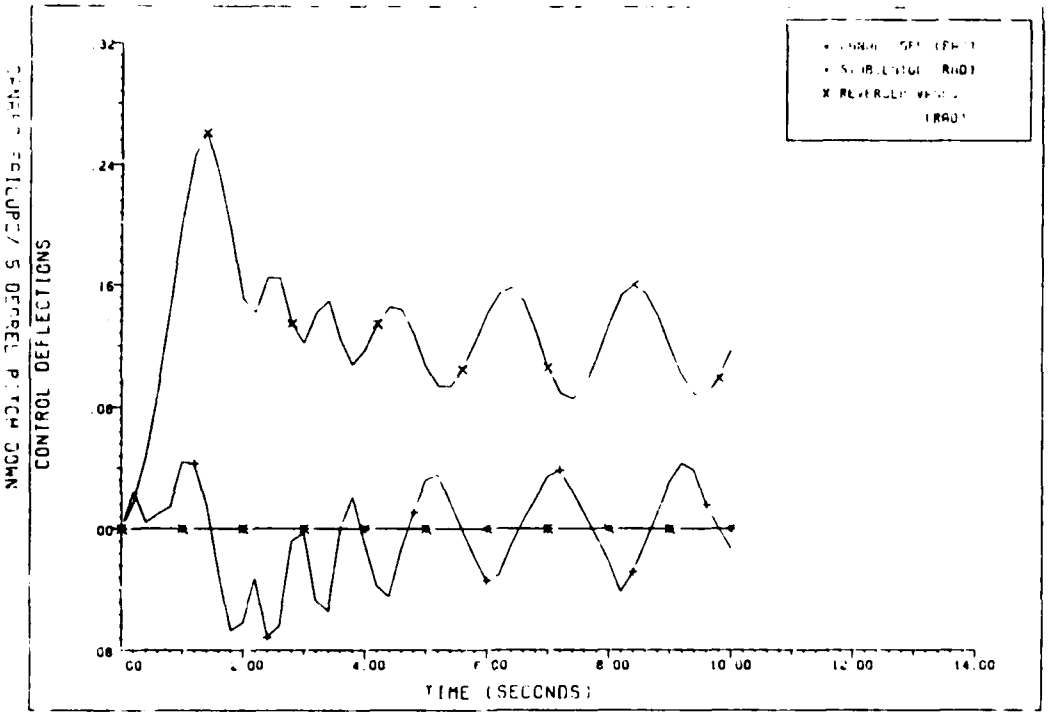


Figure 5-14. Five Degree Pitch Down with Canard Failure

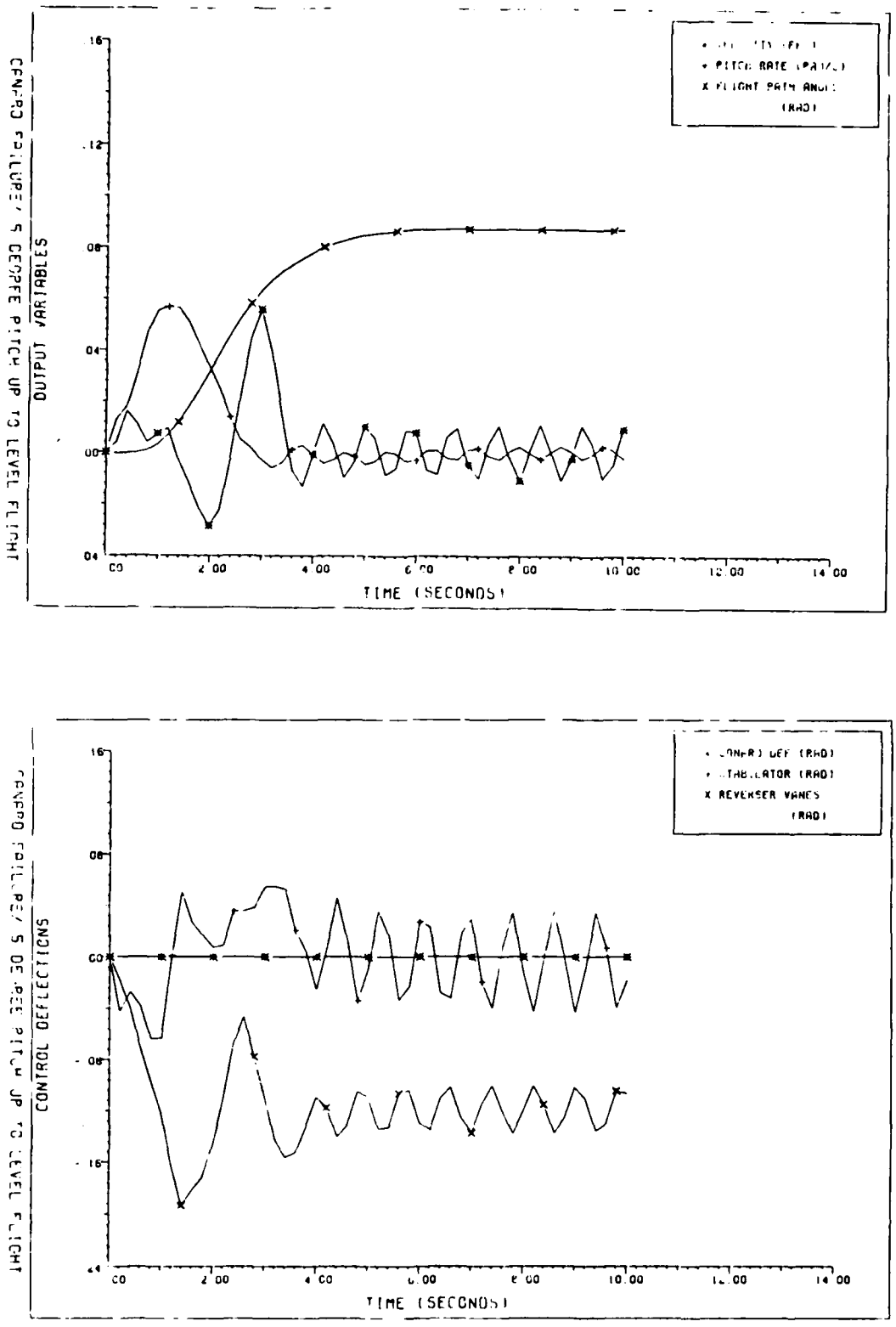
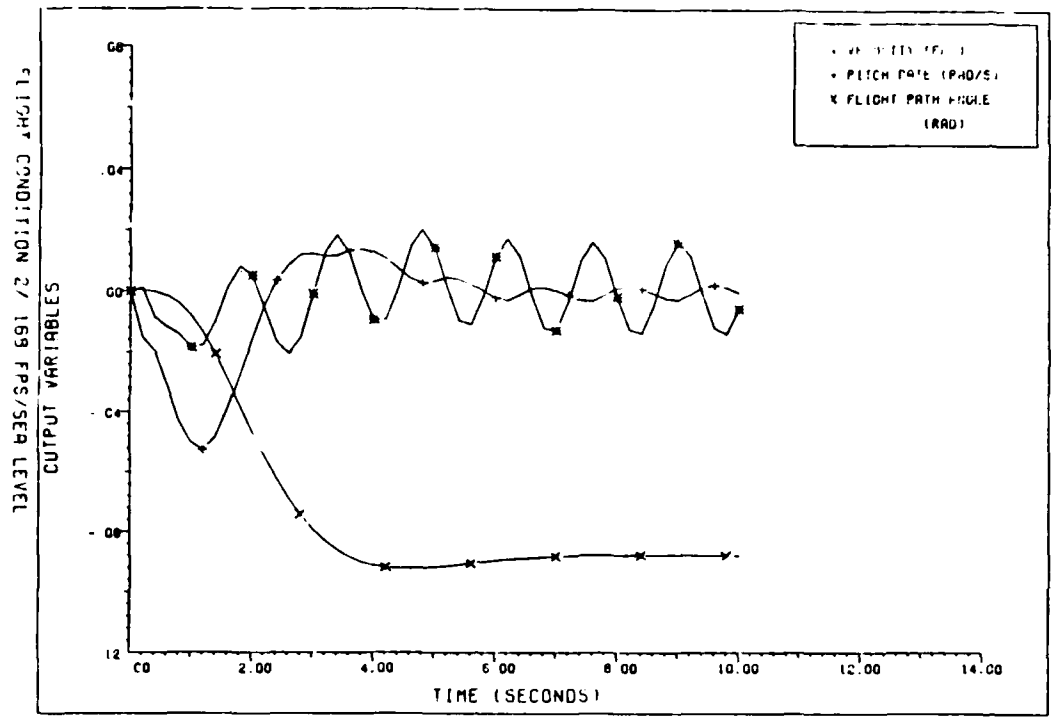


Figure 5-15. Five Degree Pitch Up to Level Flight with Canard Failure

as the controller has been evaluated against a truth model of higher order than the design model. The controller designed for a specific flight condition does not perform as well at off-design points of the flight envelope. This is not a major concern, as in practice, gain scheduling would be used for variations inflight conditions, but it is a good check for robustness. Simulations using the controller designed in this study versus the aircraft model for a slower true airspeed (V_{min}), for a higher density altitude at the original airspeed, and for an increased gross weight at the original airspeed and altitude are shown in Figures 5-16, 5-17, and 5-18, respectively. As was mentioned earlier, this controller suffers from degraded performance at the off-design conditions, which is shown by the increased variance in the velocity and pitch rate and the increased control activity required for the manuever compared to that for the baseline controller with the Kalman filter (Fig. 5-10). Again note that the undulations in velocity are very small when compared to flight velocity. Attempts will be made in the next section, via LTR, to improve this performance.

5-5 Kalman Filter Tuning

In this section the Doyle and Stein LTR technique is applied to the controller in an attempt to enhance robustness. As a slight modification to this method, q^2BVB^T



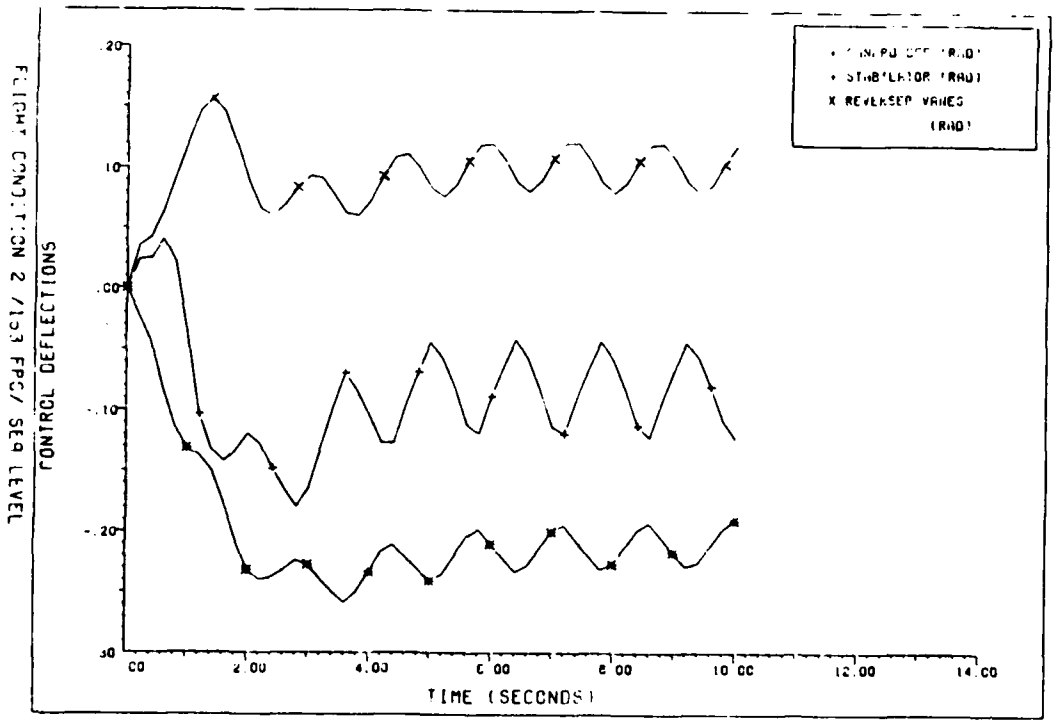
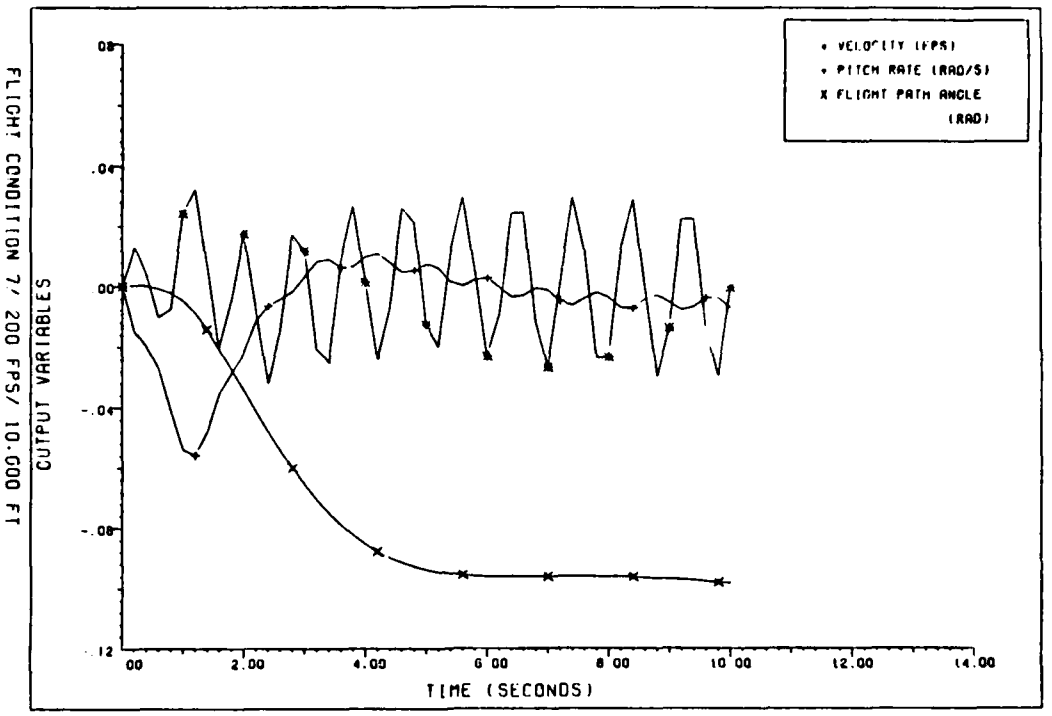


Figure 5-16. Approach at V_{min}



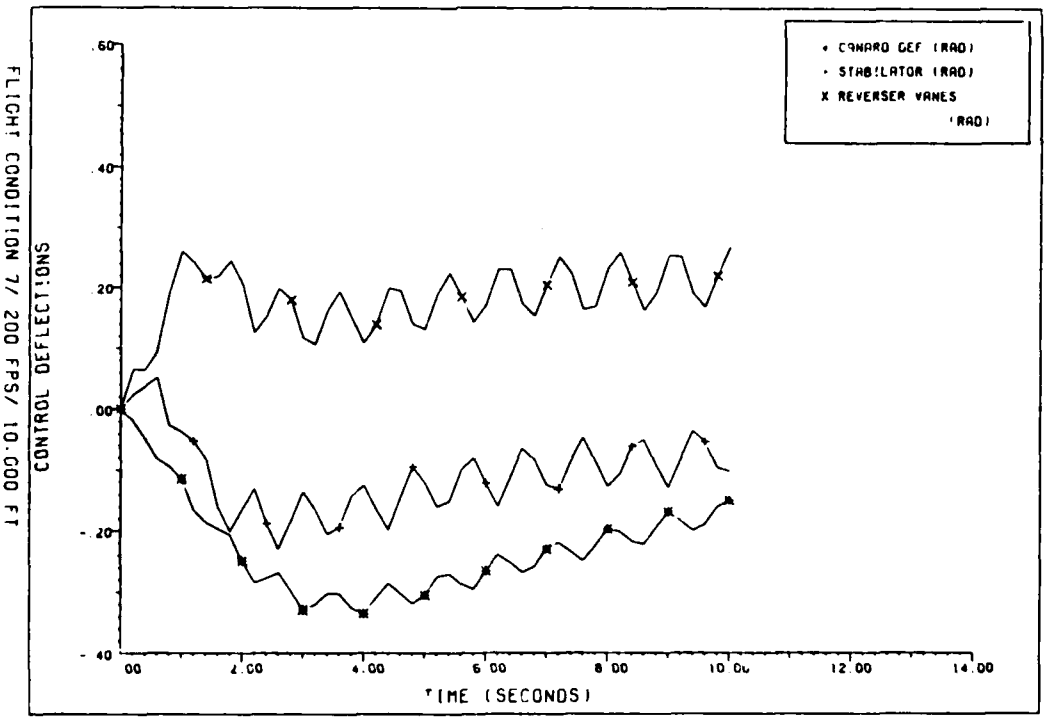
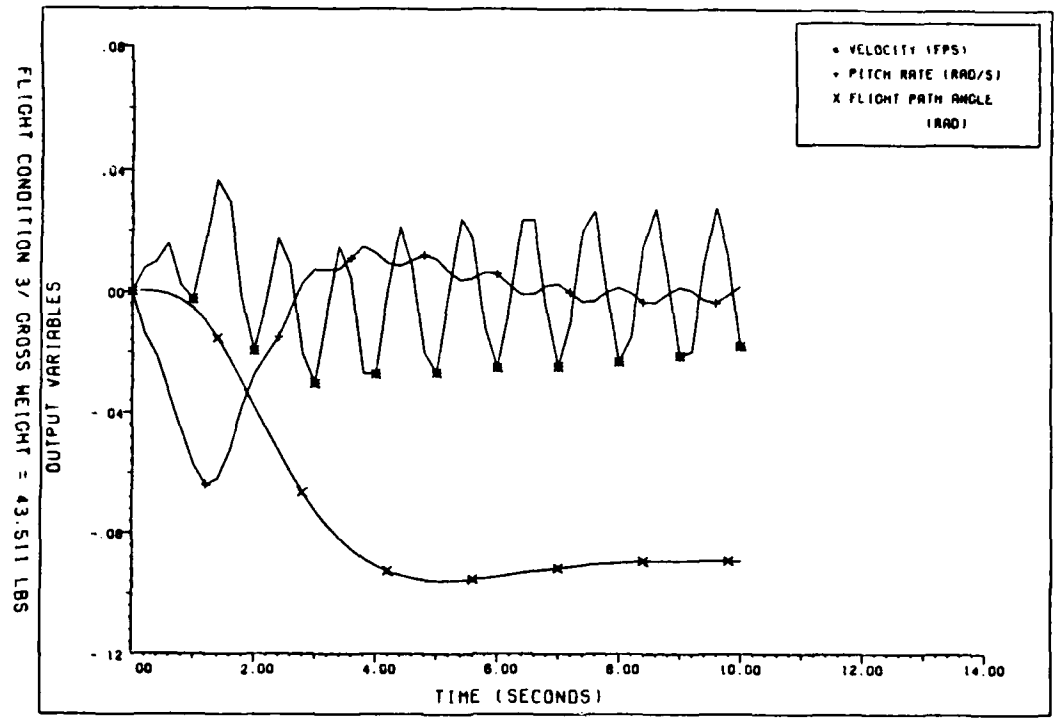


Figure 5-17. Approach with 10,000 Foot Altitude



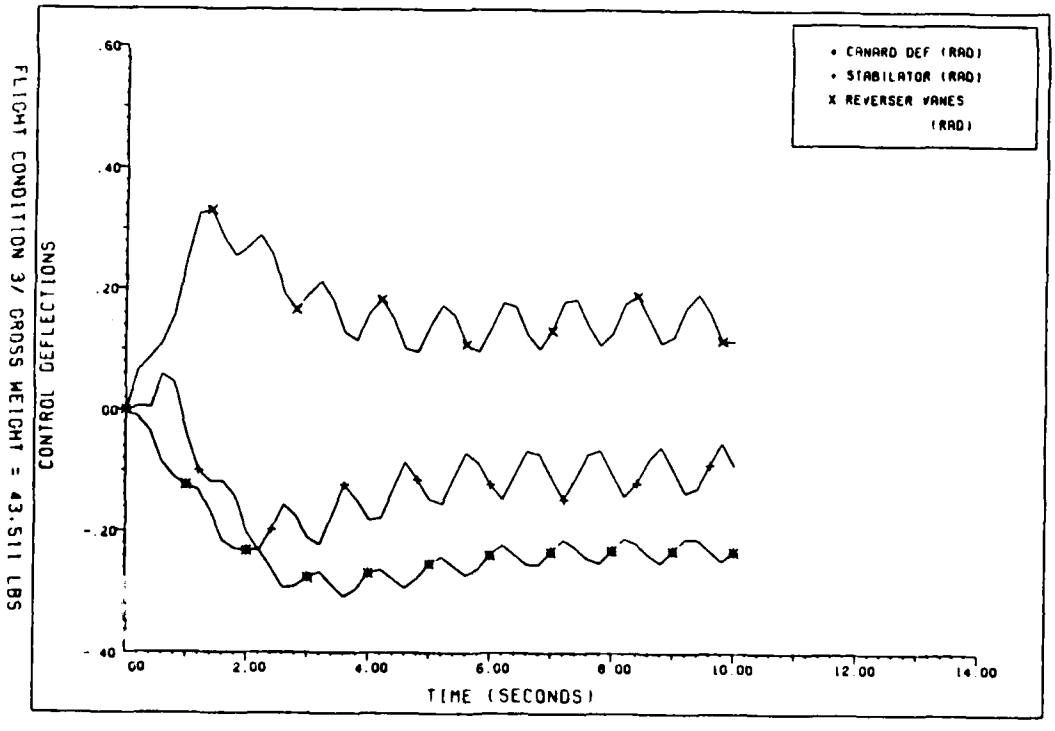


Figure 5-18. Approach with Increase Gross Weight

replaces Q_0. For this study, the q^2 term was set to .0001, and the technique was used on one of the off-design conditions and for the case with a three order-of-magnitude increase in sensor noise. The initial prospects for robustness enhancement with this method were not promising, as the system has a right-half-plane transmission zero [18] and is a sampled-data PI controller rather than a continuous-time proportional gain regulator. Separately, each of these conditions removes the guarantees associated with the robustness improvement to be gained by the Doyle and Stein technique [28].

First the controller with the filter tuned via LTR is evaluated at the flight condition for which the controller was designed. The results of this simulation, seen in Figure 5-19, show significant improvement in all channels over the baseline controller with the Kalman filter in the loop (Fig. 5-10), and approach the performance of the full-state feedback controller (Fig. 5-9). Once again, this result is expected because of the increased order of the truth model used in the evaluation, compared to the order of the design model. If this design were to be evaluated against the design model rather than the higher dimensioned truth model, LTR tuning would be off-nominal tuning at design conditions, and would be expected to cause performance degradation at the design conditions.

Next the controller is evaluated against a flight

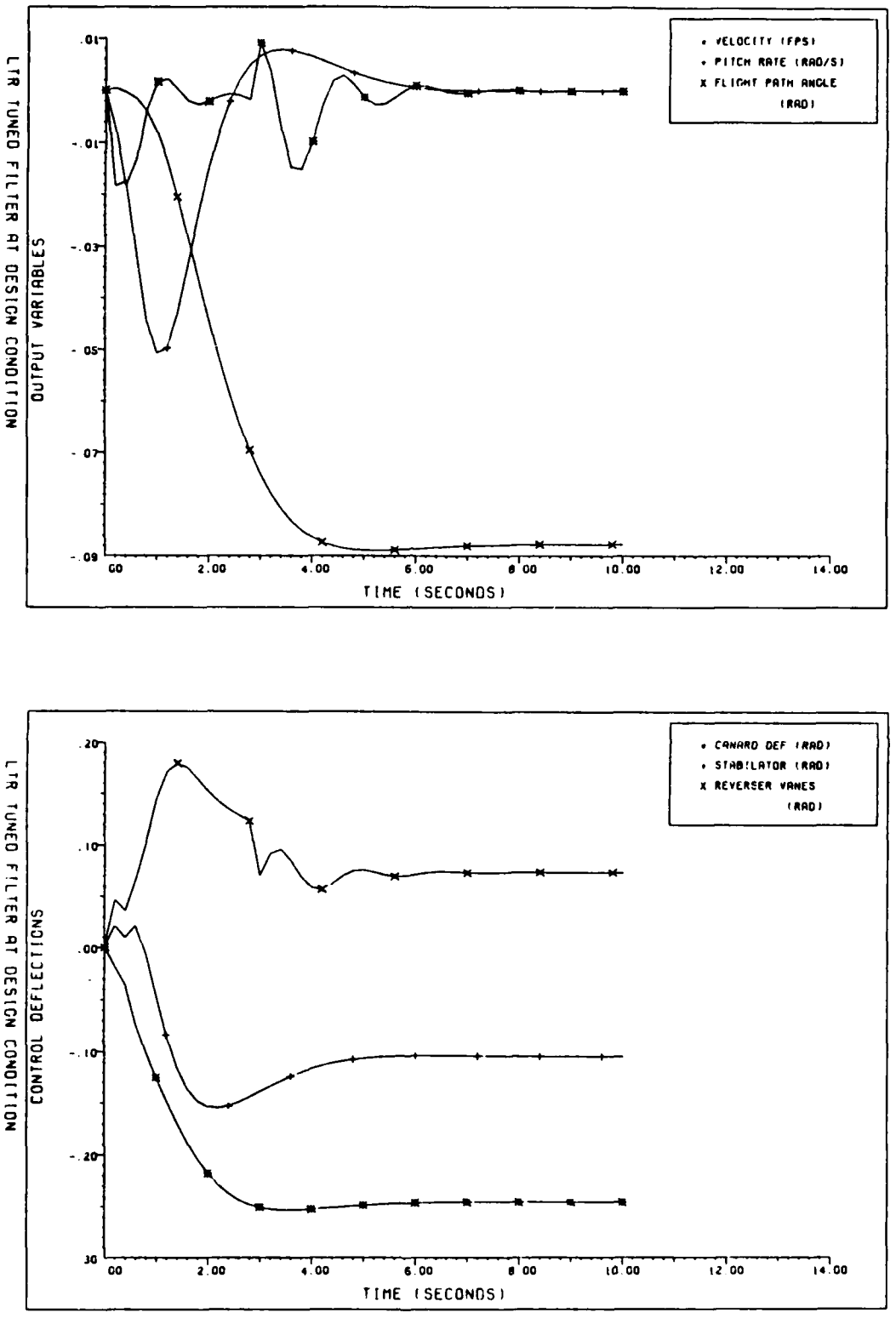
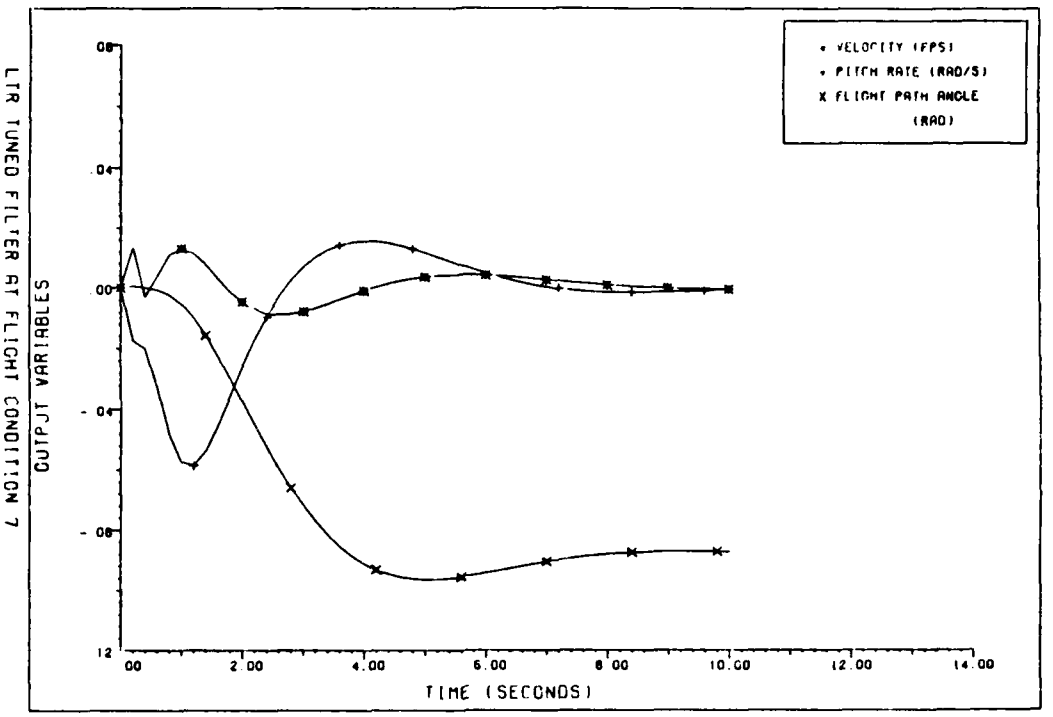


Figure 5-19. LTR-Tuned Filter/ Nominal Conditions

condition with a 10,000 foot altitude. With the original Kalman filter at the 10,000 foot altitude (Fig. 5-17), there are significant oscillations in both the velocity and pitch rate, but with the LTR-tuned filter (Fig. 5-20) these oscillations are smoothed out and there is less overshoot in the flight path angle, along with a reduction in the level and undulations of the control inputs: a definite improvement in the performance of the controller.

The controller with the LTR-tuned filter is then evaluated with the aircraft gross weight increased by approximately 10,000 pounds. The results of this simulation are shown in Figure 5-21 and, as in the preceding case, a marked improvement over the performance of the untuned filter (Fig. 5-18) is exhibited. Once again, this improvement occurs in all channels of interest, with the smoothing of the velocity, pitch rate, and control input responses the most noticeable.

In the case of a large increase in sensor noises (Figure 5-22), the controller is unstable with the original filter and is stabilized with the change to the LTR-tuned filter (Figure 5-23), although it displays very poor performance. This enhanced stability would be important, as keeping the aircraft stable in the face of a massive sensor failure would allow online adaptation to occur, or at least provide the pilot time to make an escape.



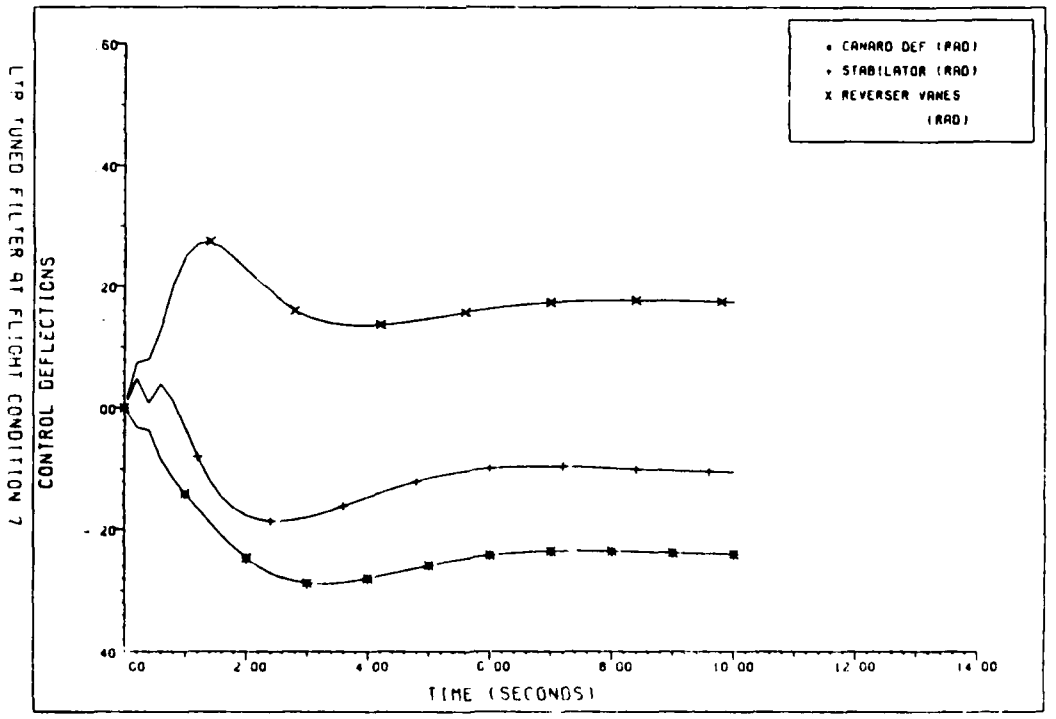
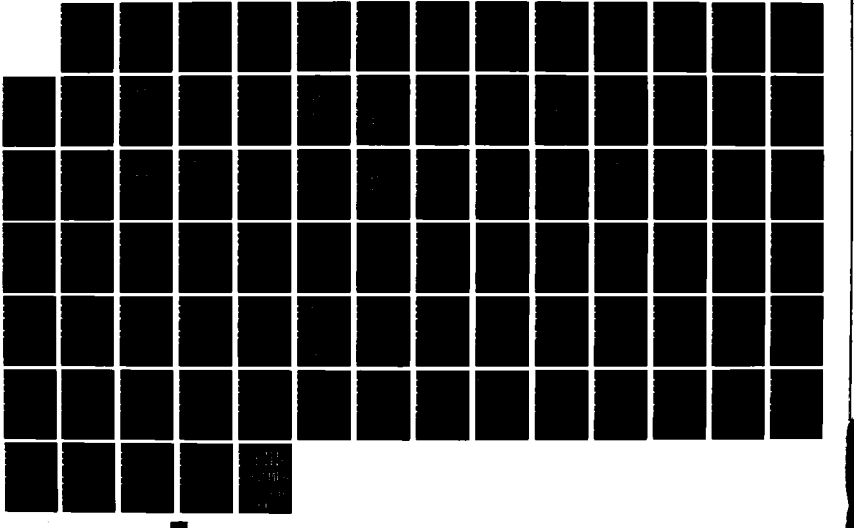
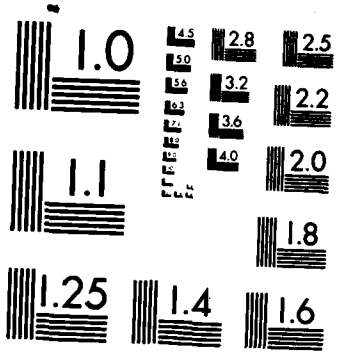


Figure 5-20. LTR-Tuned Filter/ 10,000 Foot Altitude

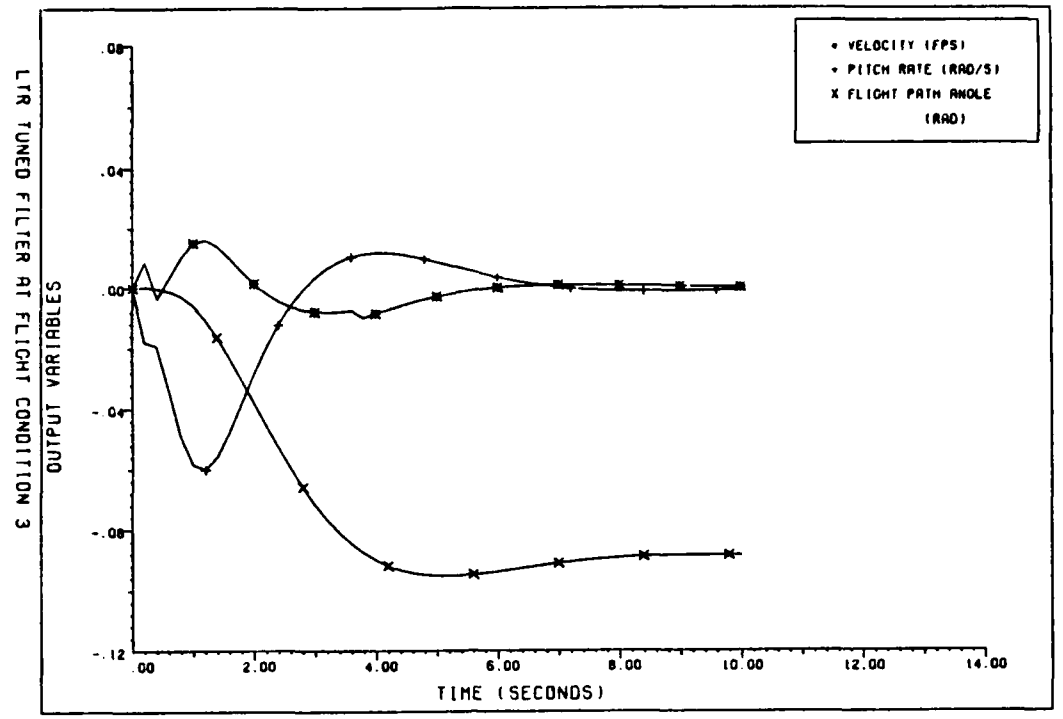
AD-A164 111 LOG/LTR DESIGN OF A ROBUST FLIGHT CONTROLLER FOR THE STOL F-15(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING G L GROSS DEC 85 2/2

UNCLASSIFIED AFIT/GAE/ENG/85D-1 F/G 17/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



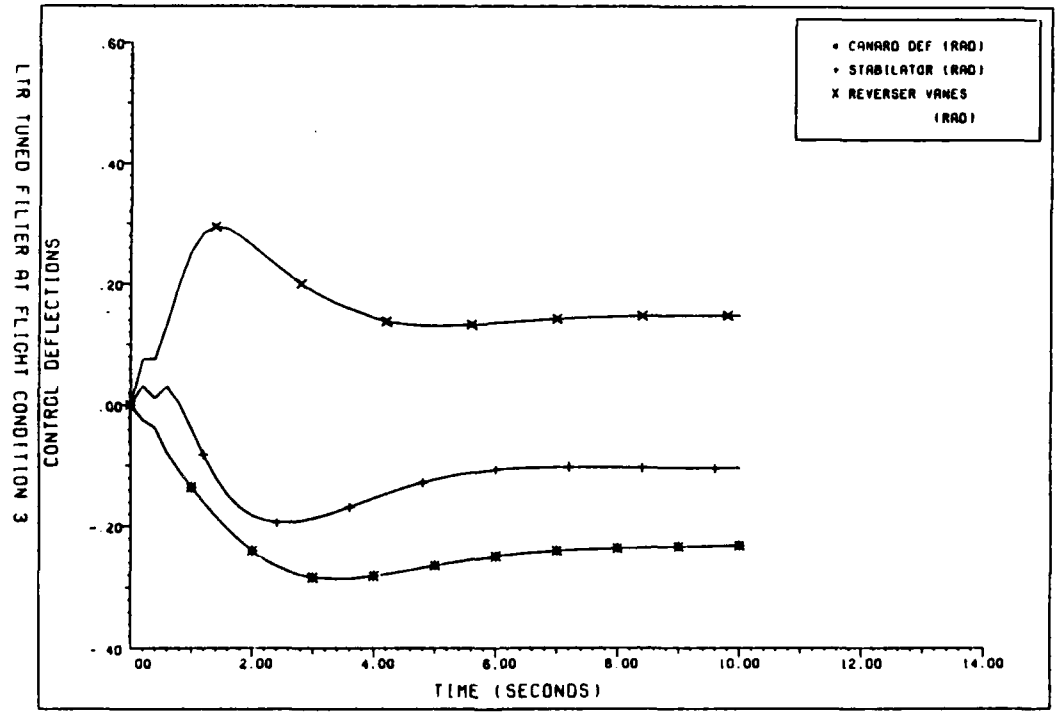
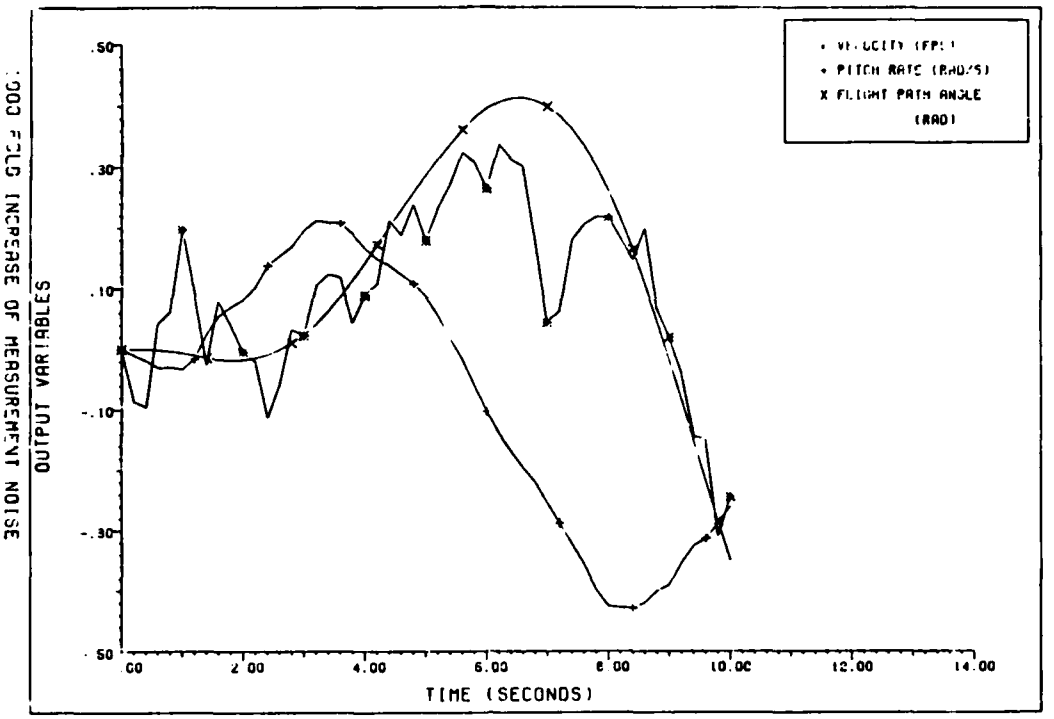


Figure 5-21. LTR-Tuned Filter/ Increased Gross Weight



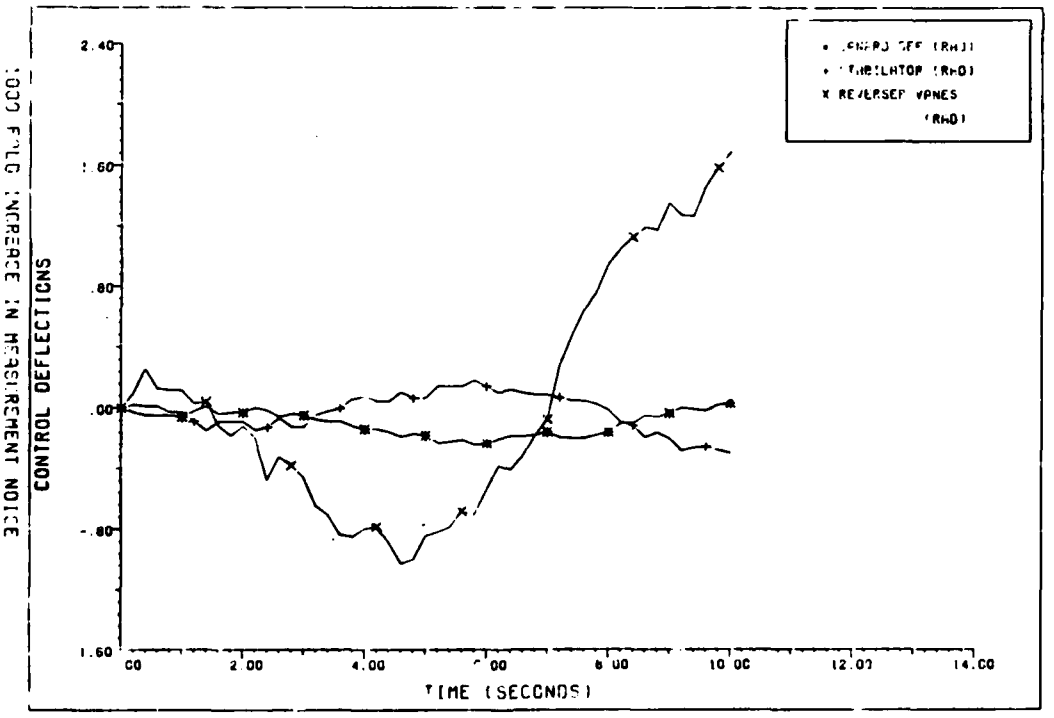


Figure 5-22. Effect of Severely Degraded Sensors

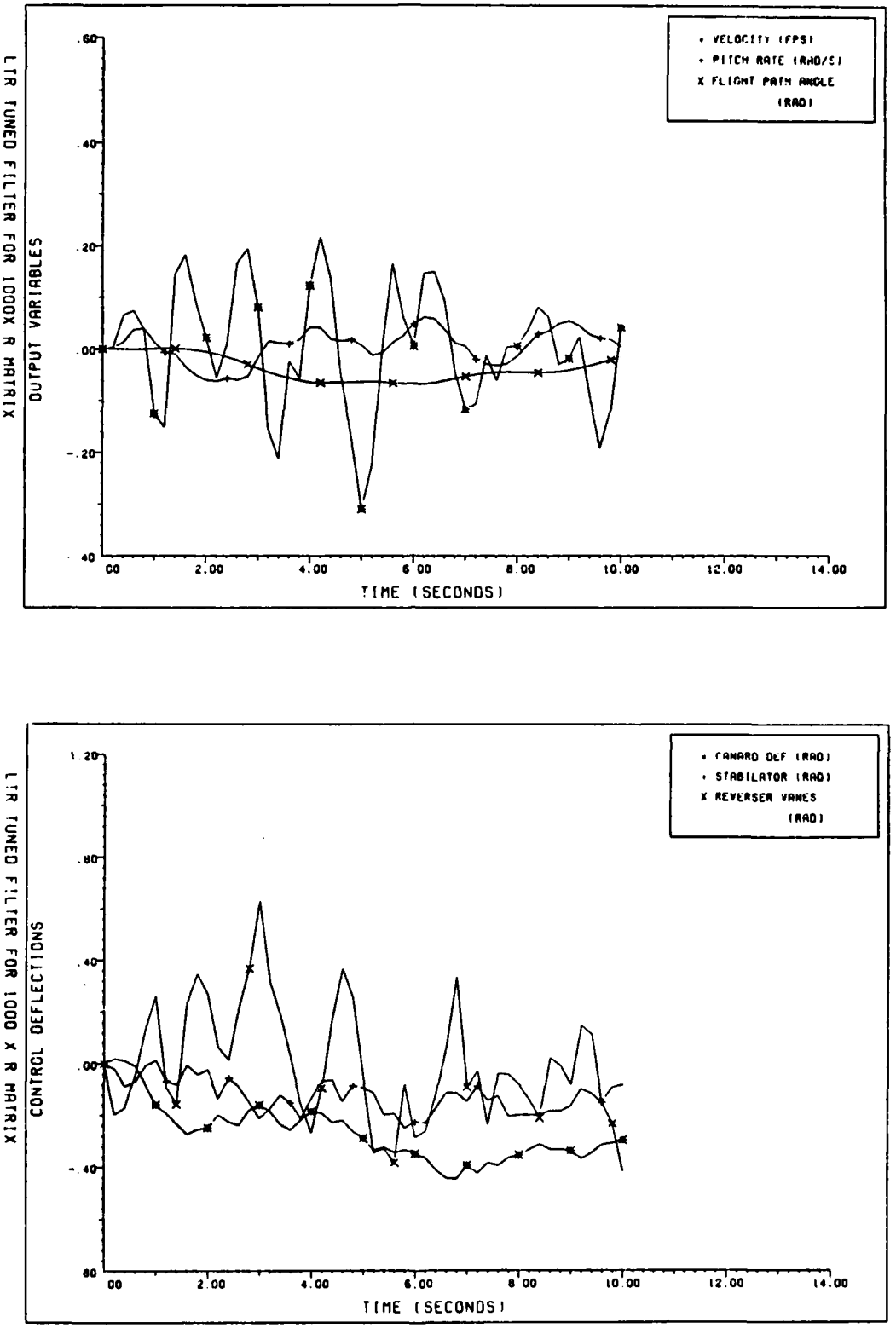


Figure 5-23. LTR-Tuned Filter/ Severely Degraded Sensors

5-6 Summary

In this chapter, the methods used for performance analysis are discussed, and then the steps taken in the development of a CGT/PI/KF controller are detailed. The resulting controller is then analyzed using the procedures from the beginning of the chapter and rather successful attempts made for robustness enhancement in the face of changing flight conditions and sensor degradation. In the next chapter, conclusions of this study and recommendations for future research will be presented.

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER STUDY

6-1 Conclusions

The goal of this study has been to design a robust LQG/LTR flight contoller for the STOL F-15 for the approach and landing phase of flight. This has been accomplished by the use of implicit model following and Loop Transfer Recovery (LTR) techniques with a Command Generator Tracker/Proportional plus Integral controller with a Kalman filter in the loop (CGT/PI/KF).

The LQG design methodology provides the designer with a systematic way to approach the design problem. Through the use of the command generator tracker (CGT), the designer can incorporate the desired handling qualities to be mimicked and the disturbances to be rejected. Implicit model following allows the designer to include information about the handling qualities directly into the performance index and also to affect the pole placement of the system. The "type-1" property of the PI controller allows for tracking of a zero input with a nonzero steady-state control, despite unmodelled constant disturbances. The Kalman filter provides estimates of the states from the noise-corrupted and incomplete outputs of sensors. The use of implicit model following allows for the design of a robust full-state feedback controller and, then LTR tuning of the filter allows for the recovery of some of that

robustness lost from the inclusion of a Kalman filter in the control loop.

One of the major benefits of LQG controller design is that the designer is provided with insights as to the next step in each iteration. These insights result from the individual weightings placed in the performance index and from the assumed characteristics of the explicit and implicit models. For example, if a certain control surface exceeds, or even approaches, a limit during an iteration, the designer knows to increase the weight on that element in the performance index on the next run.

The fact that LTR tuning yielded an improved controller was somewhat unexpected, as the system is non-minimum phase and includes a PI controller rather than a proportional gain regulator. The type of the change in performance was also not predicted: not only is the controller more robust, it also performs better at the design condition, contrary to the normal trade-off between performance at design conditions and robustness at off-design conditions.

The major drawback to the use of LQG/LTR design is the lack of an integrated design and analysis software package. Currently, three separate pieces of software must be used to design and evaluate a controller completely. Two of the codes, CGTPIF and PFEVAL, are written in FORTRAN IV and are semi-compatible. To use a file generated by CGTPIF in PFEVAL, one must exit CGTPIF, save the file,

rename the file as a data file, and then run PFEVAL. To use ODEF15, a FORTRAN V code, one must attach two separate libraries and enter the system matrices, in both continuous and discrete-time form, and the gain matrices from CGTPIF, manually. This can be time-consuming and is not efficient in the use of computer resources.

6-2 Recommendations for Further Study

The number one recommendation for future research was addressed in the previous section. An integrated design package is needed to make the study of LQG/LTR design theory more practical for the design of a CGT/PI/KF controller. This design tool could be built around one of the state-of-the-art computer aided design (CAD) packages, such as MATRIX X [18], and should be hosted on an AFIT computer, preferably the VMS VAX. Using an integrated CAD package has the advantage of built-in routines for Kalman filtering, singular value and transmission zero solutions, along with many other features of which the designer could take advantage.

Another topic for extended research would be to derive the proper LTR technique for a filter with a PI controller. Current research has derived LTR for the regulator and this study indicated that form of LTR to be beneficial to a PI controller, but there is currently no physical or mathematical foundation for the use of regulator LTR with the PI controller structure.

Finally, the applicability of structured singular values [21] to the LQG/LTR design problem should be investigated further. Unstructured singular values were previously shown to be of little benefit to design insight [24], but structured singular values may be able to bound the performance and sensitivity of the controller, and thereby provide the designer with very pertinent information for the synthesis of robust controller designs.

BIBLIOGRAPHY

1. Acker, B. H. Multivariable Output Feedback Control Law Design for the STOL F-15 in Landing Configuration. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1985.

2. ASD/ENESS, "Military Specification - Flying Qualities of Piloted Aircraft." Mil-F-8785C. Wright-Patterson Air Force Base, Ohio, 5 November 1980.

3. Barfield, A. F. Multivariable Control Laws for the AFTI/ F-16. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, July 1983.

4. Barraud, A. Y. "A Numerical Algoritm to Solve $A^TXA-X= Q$," IEEE Transactions on Automatic Control, AC-22 (5): 883-885 (October 1977).

5. Broussard, J. R. "Command Generator Tracking," TASC TIM-612-3, The Analytical Sciences Corporation, Reading, Massachusetts, March 1978.

6. Broussard, J. R. and P. W. Berry. " The Relationship Between Implicit Model Following and Eigenvalue Eigenvector Placement," IEEE Transactions on Automatic Control, AC-25 (3): 591-594 (June 1980).

7. Clough, B. Control System Design for the STOL F-15 in Landing Configuration Utilizing the Quantitative Feedback Theory Technique. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1985.

8. D'Azzo, John J., and Constantine H. Houpis. Linear Control Systems Analysis and Design (Second Edition). New York: McGraw-Hill Book Company, 1981.

9. Doyle, J. C. and G. Stein. "Robustness with Observers," IEEE Transactions on Automatic Control, AC-24 (4): 607-611 (August 1979).

10. Doyle, J. C. and G. Stein. "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," IEEE Transactions on Automatic Control, AC-26 (1): 4-16 (February 1981).

11. Floyd, R. M. Design of Advanced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Techniques, Volumes 1 and 2. M.S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1980.

12. Gilbert, E. G. "Conditions for Minimizing the Norm Sensitivity of Characteristic Roots," Conference on Information Sciences and Systems, Baltimore, Maryland. (March 1983).

13. Horowitz, I. M. and Marcel Sidi. "Synthesis of Feedback Systems with Large Plant Ignorances for Prescribed Time-domain Tolerances," International Journal of Control, 16 (2): 287-309 1972.

14. Houston, R. A. An LQG Up and Away Flight Control Design for the STOL F-15. M.S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1985.

15. Howey, J. M. Robust Flight Controllers. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1983.

16. Larimer, S. J. "TOTAL User's Manual (CAD)," Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, June 1981.

17. Lloyd, E.D. Robust Control Systems. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1981.

18. "MATRIX X User's Manual," Intergrated Systems Incorporated, Palo Alto, California, 1984.

19. Maybeck, P. S. Stochastic Models, Estimation, and Control, Volume 1. New York: Academic Press, 1979.

20. Maybeck, P. S. Stochastic Models, Estimation, and Control, Volume 3. New York: Academic Press, 1982.

21. Maybeck, P. S., R. M. Floyd, and A. Moseley. "Synthesis and Performance Evaluation Tools for CGT/PT Advanced Digital Flight Control Systems," IEEE 1983 National Aerospace and Electronics Conference (NAECON 1983), Dayton, Ohio. 1259-1266, May 1983.

22. Maybeck, P. S., W. G. Miller, and J. M. Howey. "Robustness Enhancement for LQG Digital Flight Controller Design," IEEE 1984 National Aerospace and Electronics Conference (NAECON 1984), Dayton, Ohio. 518-525, May 1984.

23. McDonnell Douglas Corporation aerodynamic data for the STOL F-15. May 1985.

24. Miller, W. G. Robust Multivariable Controller Design Via Implicit Model-Following Methods. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1983.

25. Moseley, A. Design of Advanced Digital Flight Control Systems Via Command Generator Tracker (CGT) Synthesis Techniques, Volumes 1 and 2. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1982.

26. Pope, R. E. "The Need for Multivariable Design and Analysis Techniques," AGARD GRP Lecture Series No. 117, October 1981.

27. Sheehan, K. A. Multivariable Digital Control Law Design for Enhanced Air Combar Manuevering: F-15/STOL Derivative Fighter. M. S. Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1985.

28. Stein, G. and M. Athans. "The LQG/LTR Procedure for Multivariable Feedback Control Design," Grant NGL-22-009-124. NASA Ames and Langley Research Centers. 1984.

APPENDIX A. ODEF15

A-1 Introduction

ODEF15 is an interactive computer program that was developed originally by Capt. W. G. Miller [24] and modified, as detailed in Chapter 5, for this study. ODEF15 is a nonlinear simulation tool that provides for the possible inclusion of a Kalman filter, actuator dynamics and/or saturations, rate and position limits, and the addition of anti-windup compensation. This appendix gives a brief description of the program and instructions for its use. A list of the computer code and a sample program execution are also given.

A-2 Program Execution

The original program, ODEACT, was designed only for the evaluation of a full-state feedback CGT/PI controller for the AFTI/F-16 [24]. ODEF15 has been modified to allow for the possible addition of a Kalman filter and is specific for the STOL F-15. To incorporate a Monte Carlo analysis in the program, the IMSL library is invoked, for random number generation, as well as the integration package ODE. Both of these libraries are available on the ASD CDC Cyber.

Prior to entering ODEF15, all previously developed data files that are to be used must be attached, along with the IMSL and ODE libraries. IMSL5 and ODE must be declared as libraries before execution begins. The program will prompt the user for all required entries. During the initial

execution, the user will be prompted to enter all required matrices. The user must be cautious when a Kalman filter is evaluated, as there is no built-in protection against accidentally entering a very large number of iterations in the Monte Carlo analysis. All of the time-response plots displayed at the user's terminal can be routed to a line printer, or saved to a file, upon program termination. Plot files for the CALCOMP plotter are also generated by the program and can also be saved or plotted as desired.

A-3 ODEF15 Source Listing

```
      PROGRAM ODEF15
C
C*****************************************************************
C
C SIMULATION PROGRAM TO TEST A PI  OR CGT/PI, WITH/WITHOUT KALMAN
C FILTER, BASED ON A 4-STATE MODEL OF THE STOL F-15.  OPTIONS
C INCLUDE MODIFICATION OF DYNAMICS MATRIX, USE OF 2-STATE
C ACTUATOR MODELS, APPLICATION OF RATE/POSITION LIMITS ON ACTUATORS,
C AND EMPLOYMENT OF ANTI-WINDUP COMPENSATION.  USER SUPPLIES DYNAMICS
C MATRIX, OUTPUT MATRIX, CGT COMMAND MODEL, CONTROLLER GAINS AND
C KALMAN FILTER MATRICES.
C
C DATE OF LAST REVISION:  01 NOV 85
C LIBRARIES USED:  ODE,IMSL5
C
C*****************************************************************
C
      REAL WORK(352),X(12),DX(12),OUT(51,4),T,TOUT,TSAMP,PLTVEC(260)
      REAL XTEMP(12)
      REAL AWORK(4,4),BWORK(4,4),CWORK(4,4),AM(4,4),BM(4,4),Z(3)
      REAL IPHIX(4,4)
      REAL RELERR,ABSERR,DBIM,OUT1(51,5),UOUT2(51,4),UOUT3(51,5),Y(4)
      REAL V(3),EVTMP(2)
      INTEGER NR,MMM,JJJ
      DOUBLE PRECISION DSEED
      INTEGER I,J,K,IFLAG,JFLAG,JCFLAG,NFLAG,IWORK(5),IDBIM
      INTEGER MMFLAG,IIFLAG
      COMMON/MATRIX/UO(2),A(12,12),C(4,12),KX(4,12),KZ(4,4),KXM(4,4),
     1 KXU(4,4),PHI(4,4),PHINT(4,4),CM(4,4),B(3,3),EV(2),KFLAG,MM
      COMMON/CONTRL/UNEW(4),UOLD(4),UCMD(4),UCOLD(4),XOLD(12),
     1 XMOLD(4),XM(4),MFLAG,EVA(2),
     1 H(3,4),PHIX(4,4),BD(4,3),QD(4,4),R(3,3),XHP(4),XHM(4),K(4,3)
C
      EXTERNAL F1,F3,F4,FSTOL
      CHARACTER ANSW*1,TITLE*50,DATA*6,SAVE*6,PLOT*6
C
C
C*****************************************************************
C
C INPUT SECTION.  DATA MAY BE READ IN FROM AN 'OLD' FILE, AND SAVED
C TO ANY OTHER FILE.  ONLY ONE SET OF DATA PER FILE NAME.  PLOTS ARE
C AUTOMATICALLY SAVED IN A 'PLOT FILE'.
C
C*****************************************************************
C
 20    PRINT*,' INCORPORATE KALMAN FILTER? Y/N:'
       READ(*,'(A)') ANSW
       IF (ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 20
       IF (ANSW.EQ.'Y') IIFLAG=1
       IF (ANSW.EQ.'N') IIFLAG=0
 902   PRINT*,'DATA TO BE READ FROM FILE? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 902
```

```
      IF(ANSW.EQ.'N') GO TO 30
         JCFLAG=0
         PRINT*,'ENTER NAME OF DATA FILE: '
         READ(*,'(A)')DATA
         OPEN(2,FILE=DATA,STATUS='OLD',FORM='UNFORMATTED',ERR=902)
         READ(2)((A(I,J),I=1,12),J=1,12)
      READ(2)((B(I,J),I=1,3),J=1,3)
         READ(2)((C(I,J),I=1,4),J=1,12)
         READ(2)((KX(I,J),I=1,4),J=1,12)
         READ(2)((KZ(I,J),I=1,4),J=1,4)
         READ(2)((KXU(I,J),I=1,4),J=1,4)
         READ(2)((KXM(I,J),I=1,4),J=1,4)
         READ(2)((AM(I,J),I=1,4),J=1,4)
         READ(2)((BM(I,J),I=1,4),J=1,4)
         READ(2)((CM(I,J),I=1,4),J=1,4)
      IF(IIFLAG.EQ.0) GO TO 901
      READ(2)((PHIX(I,J),I=1,4),J=1,4)
      READ(2)((BD(I,J),I=1,4),J=1,3)
      READ(2)((GD(I,J),I=1,4),J=1,4)
      READ(2)((H(I,J),I=1,3),J=1,4)
      READ(2)((K(I,J),I=1,4),J=1,3)
      READ(2)((R(I,J),I=1,3),J=1,3)
 901  CONTINUE
         REWIND(2)
         CLOSE(2)
      GO TO 140
C
C FOR KEYBOARD INPUT, ONLY NON-ZERO MATRIX ELEMENTS ARE REQUIRED.
C NO NON-ZERO ENTRIES SHOULD BE MADE FOR COLUMNS 5,6,7,9,10 OR 11
C OF A OR KX, BUT NO PROTECTION PROVIDED AGAINST DOING SO.
C
 30   DO 50 I=1,12
         DO 50 J=1,12
            A(I,J)=0.0
 50   CONTINUE
      DO 42 I=1,3
        DO 42 J=1,3
              R(I,J)=0.0
           B(I,J)=0.0
42    CONTINUE
      DO 64 I=1,12
         DO 60 J=1,4
            KX(J,I)=0.0
 60      CONTINUE
         DO 64 L=1,4
            C(L,I)=0.0
 64   CONTINUE
      DO 66 I=1,4
         DO 66 J=1,4
                PHIX(I,J)=0.0
            KZ(I,J)=0.0
            KXU(I,J)=0.0
            KXM(I,J)=0.0
```

```
 66   CONTINUE
      DO 70 I=1,4
         DO 70 J=1,4
            AM(I,J)=0
            BM(I,J)=0
            CM(I,J)=0
            QD(I,J)=0.0
 70   CONTINUE
      DO 903 I=1,4
         DO 903 J=1,3
            H(J,I)=0.0
            K(I,J)=0.0
            BD(I,J)=0.0
 903  CONTINUE
      JFLAG=0
 72   PRINT*,'ENTER DYNAMICS MATRIX: '
      CALL EDIT(A,12,12)
      IF(JFLAG.NE.0) GO TO 140
 74   PRINT*,'ENTER CONTROL MATRIX: '
      CALL EDIT(B,3,3)
      IF(JFLAG.NE.0) GO TO 140
 76   PRINT*,'ENTER OUTPUT MATRIX: '
      CALL EDIT(C,4,12)
      IF(JFLAG.NE.0) GO TO 140
 78   PRINT*,'ENTER KX MATRIX: '
      CALL EDIT(KX,4,12)
      IF(JFLAG.NE.0) GO TO 140
 80   PRINT*,'ENTER KZ MATRIX: '
      CALL EDIT(KZ,4,4)
      IF(JFLAG.NE.0) GO TO 140
 82   PRINT*,'ENTER KXM MATRIX: '
      CALL EDIT(KXM,4,4)
      IF(JFLAG.NE.0) GO TO 140
 84   PRINT*,'ENTER KXU MATRIX: '
      CALL EDIT(KXU,4,4)
      IF(JFLAG.NE.0) GO TO 140
 86   PRINT*,'ENTER MODEL DYNAMICS MATRIX: '
      JCFLAG=0
      CALL EDIT(AM,4,4)
      IF(JFLAG.NE.0) GO TO 140
 88   PRINT*,'ENTER MODEL CONTROL MATRIX: '
      JCFLAG=0
      CALL EDIT(BM,4,4)
      IF(JFLAG.NE.0) GO TO 140
 90   PRINT*,'ENTER MODEL OUTPUT MATRIX: '
      CALL EDIT(CM,4,4)
      IF(JFLAG.NE.0) GO TO 140
      IF(IIFLAG.EQ.0) GO TO 140
 904  PRINT*,'ENTER STATE TRANSITION MATRIX:'
      CALL EDIT(PHIX,4,4)
      IF(JFLAG.NE.0) GO TO 140
 905  PRINT*,'ENTER DISCRETE TIME INPUT MATRIX'
      CALL EDIT(BD,4,3)
```

```
      IF(JFLAG.NE.0) GO TO 140
  906 PRINT*,'ENTER DISCRETE TIME COVARIANCE MATRIX:'
      CALL EDIT(QD,4,4)
      IF(JFLAG.NE.0) GO TO 140
  907  PRINT*,'ENTER MEASUREMENT MATRIX:'
      CALL EDIT(H,3,4)
      IF(JFLAG.NE.0) GO TO 140
  908  PRINT*,'ENTER KALMAN FILTER GAINS:'
      CALL EDIT(K,4,3)
      IF(JFLAG.NE.0) GO TO 140
  909 PRINT*,'ENTER MEASUREMENT NOISE COV. MATRIX'
      CALL EDIT(R,3,3)
      IF(JFLAG.NE.0) GO TO 140
  140 PRINT*,'ANY CHANGES TO MATRICES? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 140
  142 IF(ANSW.EQ.'Y') THEN
         PRINT*,'1=A     2=C    3=KX    4=KZ    5=KXM    6=KXU'
         PRINT*,'7=AM    8=BM    9=CM  10=B '
      PRINT*,'11=PHIX   12=BD   13=QD   14=H   15=K  16=R'
      PRINT*,' ENTER CHOICE:'
         READ*,JFLAG
         GO TO (72,76,78,80,82,84,86,88,90,74,904,905,
     1 906,907,908,909) JFLAG
      ELSE
         JFLAG=0
      END IF
  150 PRINT*,'WRITE DATA TO OUTPUT FILE? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 150
      IF(ANSW.EQ.'Y') THEN
         PRINT*,'ENTER NAME OF OUTPUT FILE: '
         READ(*,'(A)')SAVE
         OPEN(3,FILE=SAVE,FORM='UNFORMATTED',ERR=80)
         WRITE(3)((A(I,J),I=1,12),J=1,12)
      WRITE(3)((B(I,J),I=1,3),J=1,3)
         WRITE(3)((C(I,J),I=1,4),J=1,12)
         WRITE(3)((KX(I,J),I=1,4),J=1,12)
         WRITE(3)((KZ(I,J),I=1,4),J=1,4)
         WRITE(3)((KXU(I,J),I=1,4),J=1,4)
         WRITE(3)((KXM(I,J),I=1,4),J=1,4)
         WRITE(3)((AM(I,J),I=1,4),J=1,4)
         WRITE(3)((BM(I,J),I=1,4),J=1,4)
         WRITE(3)((CM(I,J),I=1,4),J=1,4)
      IF(IIFLAG.EQ.0) GO TO 910
         WRITE(3)((PHIX(I,J),I=1,4),J=1,4)
         WRITE(3)((BD(I,J),I=1,4),J=1,3)
         WRITE(3)((QD(I,J),I=1,4),J=1,4)
      WRITE(3)((H(I,J),I=1,3),J=1,4)
         WRITE(3)((K(I,J),I=1,4),J=1,3)
         WRITE(3)((R(I,J),I=1,3),J=1,3)
  910 CONTINUE
         ENDFILE(3)
```

```
         REWIND(3)
         CLOSE(3)
      END IF
C
C*****************************************************************
C
C NOW SET UP CONDITIONS FOR CALLING ODE.  ALL INITIAL CONDITIONS
C ARE ZERO UNLESS CHANGED BY USER INPUT.
C
C*****************************************************************
C
      IF(JCFLAG.EQ.0) THEN
         PRINT*,'ENTER SAMPLING TIME: '
         READ*,TSAMP
         CALL DSCRT(AM,4,TSAMP,PHI,PHINT,30,AWORK,BWORK,CWORK)
         CALL MATML(PHINT,BM,AWORK,4,4,4)
         CALL COPYMT(AWORK,PHINT,4,4)
      PRINT*,'   '
         JCFLAG=1
      END IF
      PRINT*,'ENTER CANARD TRIM ANGLE OF ATTACK'
      READ*,EV(1)
      PRINT*,'ENTER STABILATOR TRIM ANGLE OF ATTACK IN RADIANS '
      READ*,EV(2)
      EVTMP(1)=EV(1)
      EVTMP(2)=EV(2)
 154  PRINT*,' RATE/POSITION LIMITS? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 154
      IF(ANSW.EQ.'Y') MFLAG=1
      IF(ANSW.EQ.'N') MFLAG=0
 156  PRINT*,' ANTI-WINDUP COMPENSATION? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 156
      IF(ANSW.EQ.'Y') NFLAG=1
      IF(ANSW.EQ.'N') NFLAG=0
 222    PRINT*,'EMPLOY ACTUATOR DYNAMICS? Y/N: '
       READ(*,'(A)')ANSW
       IF(ANSW.NE.'Y'.AND.ANSW.NE.'N')GO TO 222
       IF(ANSW.EQ.'Y')MM=0
       IF(ANSW.EQ.'N')MM=1
 158  PRINT*,'ENTER DESIRED RESPONSE DURATION: '
      READ*,DSIM
      IF(DSIM.LT.0.1) GO TO 158
      IDSIM=INT(DSIM/(50.0*TSAMP)+.99)
 160  DO 170 I=1,12
         X(I)=0.0
         XOLD(I)=0.0
 170  CONTINUE
      EVA(1)=0.0
      EVA(2)=0.0
      DO 172 I=1,4
         UOLD(I)=0.0
```

```
          Y(I)=0.0
        XHP(I)=0.0
        XHM(I)=0.0
           UNEW(I)=0.0
  172   CONTINUE
        DO 175 I=1,4
           UCMD(I)=0.0
           UCOLD(I)=0.0
           XM(I)=0.0
           XMOLD(I)=0.0
        Y(I)=0.0
  175   CONTINUE
  180   PRINT*,'ENTER I AND X(I); 0,0 TO TERMINATE: '
  190   READ*,IIII,EL
        IF(IIII.LE.12.AND.IIII.GE.1) THEN
           X(IIII)=EL
           GO TO 190
        ELSE IF(IIII.EQ.0) THEN
           GO TO 200
        ELSE
           PRINT*,'SUBSCRIPT OUT OF RANGE'
           GO TO 180
        END IF
  200    PRINT*,' SELECT COMMAND INPUT & STEP MAGNITUDE:'
         READ*,IK,ELL
         IF(IK.LE.3.AND.IK.GE.1)THEN
         UCMD(IK)=ELL
         ELSE
         PRINT*,' SUBSCRIPT OUT OF RANGE'
         GO TO 200
         END IF
        T=0.0
        TOUT=0.0
        IFLAG=-1
        RELERR=1.E-08
        ABSERR=1.E-07
        UO(1)=0.0
        UO(2)=0.0
        IF(IIFLAG.EQ.1)THEN
        PRINT*,'ENTER SOURCE OF OUTPUT MEASUREMENTS'
        PRINT*,'1=SYSTEM STATES   2=OUTPUT VARIABLES'
        READ*,NN
        IF(NN.EQ.1)THEN
  913   PRINT*,'ENTER STATES TO BE MEASURED (3)'
        READ*,L,M,N
        IF((L.GT.12).OR.(L.LT.1)) GO TO 913
        IF((M.GT.12).OR.(M.LT.1)) GO TO 913
        IF((N.GT.12).OR.(M.LT.1)) GO TO 913
        ELSE
  914   PRINT*,'ENTER OUTPUT VARIABLES TO BE MEASURED (3)'
        READ*,L,M,N
        IF((L.GT.3).OR.(L.LT.1)) GO TO 914
        IF((M.GT.3).OR.(M.LT.1)) GO TO 914
```

```
      IF((N.GT.3).OR.(N.LT.1)) GO TO 914
      END IF
      END IF
      XITER=1.
      IF(IIFLAG.EQ.0) GO TO 920
951   PRINT*,'ENTER # OF ITERATIONS FOR FILTER AVERAGE'
      READ*,XITER
      NR=3
      PRINT*,'SELECT SEED VALUE FOR RANDOM # GENERATION'
      READ*,DSEED
937   PRINT*,'CONTROL BASED ON XHAT+ OR XHAT-?'
      PRINT*,'XHAT+=1      XHAT-=2'
      READ*,JMFLAG
      IF((JMFLAG.GT.2).OR.(JMFLAG.LT.1))GO TO 937
920   CONTINUE
      DO 750 II=1,51
      OUT(II,1)=0.
      OUT1(II,1)=0.
      UOUT2(II,1)=0.
      UOUT3(II,1)=0.
      OUT(II,2)=0.
      OUT(II,3)=0.
      OUT(II,4)=0.
      OUT1(II,2)=0.
      OUT1(II,3)=0.
      OUT1(II,4)=0.
      OUT1(II,5)=0.
      UOUT2(II,2)=0.
      UOUT2(II,3)=0.
      UOUT2(II,4)=0.
      UOUT3(II,2)=0.
      UOUT3(II,3)=0.
      UOUT3(II,4)=0.
      UOUT3(II,5)=0.
750   CONTINUE
      IF(IIFLAG.EQ.0) GO TO 1040
      DO 780 IJK=1,XITER
      T=0.0
      TOUT=0.0
      IFLAG=-1
      RELERR=1.E-08
      ABSERR=1.E-07
      UO(1)=0.0
      UO(2)=0.0
      DO 1010 I=1,4
      XM(I)=0.0
      UOLD(I)=0.0
      UNEW(I)=0.0
      UCMD(I)=0.
      UCOLD(I)=0.
      XMOLD(I)=0.
      Y(I)=0.
      XHP(I)=0.
```

```
        XMM(I)=0.
 1010  CONTINUE
       DO 1020 I=1,12
       XTEMP(I)=0.
       X(I)=0.
       XOLD(I)=0.
 1020  CONTINUE
       X(IIII)=EL
       UCMD(IK)=ELL
       EVA(1)=0.0
       EVA(2)=0.0
       EV(1)=EVTMP(1)
       EV(2)=EVTMP(2)
 1040  CONTINUE
       DO 300 I=IDSIM+1,51*IDSIM+1
       CALL ERXSET(300,0)
       CALL MATML(C,X,Y,4,12,1)
          IF(IDSIM.NE.1.AND.MOD(I,IDSIM).EQ.1) THEN
             J=INT(I/IDSIM)
            OUT(J,1)=TOUT+OUT(J,1)
            OUT(J,2)=Y(1)+OUT(J,2)
            OUT(J,3)=Y(2)+OUT(J,3)
            OUT(J,4)=Y(3)+OUT(J,4)
C
            OUT1(J,1)=TOUT+OUT1(J,1)
            OUT1(J,2)=UNEW(1)+OUT1(J,2)
            OUT1(J,3)=UNEW(2)+OUT1(J,3)
            OUT1(J,4)=UNEW(3)+OUT1(J,4)
            OUT1(J,5)=UNEW(4)+OUT1(J,5)
C
            UOUT2(J,1)=TOUT+UOUT2(J,1)
            UOUT2(J,2)=X(5)+UOUT2(J,2)
            UOUT2(J,3)=X(7)+UOUT2(J,3)
            UOUT2(J,4)=X(9)+UOUT2(J,4)
C
C
            UOUT3(J,1)=TOUT+UOUT3(J,1)
            UOUT3(J,2)=X(1)+UOUT3(J,2)
            UOUT3(J,3)=X(2)+UOUT3(J,3)
            UOUT3(J,4)=X(3)+UOUT3(J,4)
            UOUT3(J,5)=X(4)+UOUT3(J,5)
           ELSE IF(IDSIM.EQ.1) THEN
              J=I-IDSIM
              OUT(J,1)=TOUT+OUT(J,1)
              OUT(J,2)=Y(1)+OUT(J,2)
              OUT(J,3)=Y(2)+OUT(J,3)
              OUT(J,4)=Y(3)+OUT(J,4)
C
              OUT1(J,1)=TOUT+OUT1(J,1)
              OUT1(J,2)=UNEW(1)+OUT1(J,2)
              OUT1(J,3)=UNEW(2)+OUT1(J,3)
              OUT1(J,4)=UNEW(3)+OUT1(J,4)
              OUT1(J,5)=UNEW(4)+OUT1(J,5)
```

```
C
                UOUT2(J,1)=TOUT+UOUT2(J,1)
                UOUT2(J,2)=X(5)+UOUT2(J,2)
                UOUT2(J,3)=X(7)+UOUT2(J,3)
                UOUT2(J,4)=X(9)+UOUT2(J,4)
C
                UOUT3(J,1)=TOUT+UOUT3(J,1)
                UOUT3(J,2)=X(1)+UOUT3(J,2)
                UOUT3(J,3)=X(2)+UOUT3(J,3)
                UOUT3(J,4)=X(3)+UOUT3(J,4)
                UOUT3(J,5)=X(4)+UOUT3(J,5)
             END IF
             TOUT=TOUT+TSAMP
          IF(IIFLAG.EQ.1)THEN
          CALL GGNML(DSEED,NR,V)
          V(1)=R(1,1)*V(1)
          V(2)=R(2,2)*V(2)
          V(3)=R(3,3)*V(3)
          IF(NN.EQ.1)THEN
          Z(1)=X(L)+V(1)
          Z(2)=X(M)+V(2)
          Z(3)=X(N)+V(3)
          ELSE
          Z(1)=Y(L)+V(1)
          Z(2)=Y(M)+V(2)
          Z(3)=Y(N)+V(3)
          END IF
          CALL KFILT(Z)
          IF(JMFLAG.EQ.1) THEN
          DO 915 J=1,4
          XTEMP(J)=XHP(J)
          DO 915 KK=5,12
          XTEMP(KK)=0.0
  915     CONTINUE
          ELSE
          DO 980 J=1,4
          XTEMP(J)=XHM(J)
          DO 980 KK=5,12
          XTEMP(KK)=0.0
  980     CONTINUE
          END IF
             CALL GCSTAR(XTEMP,NFLAG)
             DO 250 J=1,12
                XOLD(J)=XTEMP(J)
  250        CONTINUE
             DO 260 J=1,4
                UOLD(J)=UNEW(J)
  260        CONTINUE
             DO 262 J=1,4
                UCOLD(J)=UCMD(J)
                XMOLD(J)=XM(J)
  262        CONTINUE
          ELSE
```

```
      CALL SCSTAR(X,NFLAG)
      DO 1060 JM=1,12
      XOLD(JM)=X(JM)
1060  CONTINUE
      DO 1070 JM=1,4
      XMOLD(JM)=XM(JM)
      UCOLD(JM)=UCMD(JM)
      UOLD(JM)=UNEW(JM)
1070  CONTINUE
      END IF
            CALL ODE(FSTOL,12,X,T,TOUT,RELERR,ABSERR,IFLAG,WORK,IWORK)
         T=TOUT
         IF(IFLAG.NE.2) THEN
            PRINT'(" IFLAG = ",I2)',IFLAG
         ELSE
            IFLAG=-2
         END IF
 300  CONTINUE
780   CONTINUE
      IF(IIFLAG.EQ.0)GO TO 1310
      DO 755 I=1,51
      OUT(I,1)=OUT(I,1)/XITER
      OUT(I,2)=OUT(I,2)/XITER
      OUT(I,3)=OUT(I,3)/XITER
      OUT(I,4)=OUT(I,4)/XITER
      OUT1(I,1)=OUT1(I,1)/XITER
      OUT1(I,2)=OUT1(I,2)/XITER
      OUT1(I,3)=OUT1(I,3)/XITER
      OUT1(I,4)=OUT1(I,4)/XITER
      OUT1(I,5)=OUT1(I,5)/XITER
      UOUT2(I,1)=UOUT2(I,1)/XITER
      UOUT2(I,2)=UOUT2(I,2)/XITER
      UOUT2(I,3)=UOUT2(I,3)/XITER
      UOUT2(I,4)=UOUT2(I,4)/XITER
      UOUT3(I,1)=UOUT3(I,1)/XITER
      UOUT3(I,2)=UOUT3(I,2)/XITER
      UOUT3(I,3)=UOUT3(I,3)/XITER
      UOUT3(I,4)=UOUT3(I,4)/XITER
      UOUT3(I,5)=UOUT3(I,5)/XITER
755   CONTINUE
1310  CONTINUE
999   PRINT*,' 1=OUTPUT VARIABLES  2=CONTROL INPUTS '
      PRINT*,' 3=CONTROL DEFLECTIONS  4=SYSTEM STATES'
      READ*, MMFLAG
      GO TO (1000,1100,1200,1300) MMFLAG
1000  CALL SETPLT(OUT,51,5,PLTVEC)
      PRINT*,' +---------------ENTER TITLE FOR PLOT-------------+'
      PRINT*
      READ(*,'(A)')TITLE
      CALL PLOTLP(PLTVEC,51,3,-1,1,0,TITLE)
      GO TO 1400
1100  CALL SETPLT(OUT1,51,5,PLTVEC)
      PRINT*,' +---------------ENTER TITLE FOR PLOT-------------+'
```

```
      PRINT*
      READ(*,'(A)')TITLE
      CALL PLOTLP(PLTVEC,51,4,-1,1,0,TITLE)
      GO TO 1400
1200  CALL SETPLT(UOUT2,51,5,PLTVEC)
      PRINT*,' +---------------ENTER TITLE FOR PLOT-----------+'
      PRINT*
      READ(*,'(A)')TITLE
      CALL PLOTLP(PLTVEC,51,3,-1,1,0,TITLE)
      GO TO 1400
1300  CALL SETPLT(UOUT3,51,5,PLTVEC)
      PRINT*,' +---------------ENTER TITLE FOR PLOT-----------+'
      PRINT*
      READ(*,'(A)')TITLE
      CALL PLOTLP(PLTVEC,51,4,-1,1,0,TITLE)
1400  PRINT*,' MORE OUTPUT PLOTS?'
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 1500
      IF(ANSW.EQ.'Y') GO TO 999
1500  CONTINUE
 3214 PRINT*,'INPUT NAME FOR CALCOMP PLOT OF OUTPUT'
      READ(*,'(A)')PLOT
      OPEN(5,FILE=PLOT,STATUS='NEW',FORM='FORMATTED',ERR=3214)
      WRITE(5,FMT='(4E20.5)')((OUT(J,I),I=1,4),J=1,51)
      ENDFILE(5)
      REWIND(5)
      CLOSE(5)
 3215  PRINT*,'INPUT NAME FOR CALCOMP PLOT OF CTRL. DEF.'
       READ(*,'(A)')PLOT
       OPEN(6,FILE=PLOT,STATUS='NEW',FORM='FORMATTED',ERR=3215)
       WRITE(6,FMT='(4E20.5)')((UOUT2(J,I),I=1,4),J=1,51)
       ENDFILE(6)
       REWIND(6)
       CLOSE(6)
 525  PRINT*,'CHANGE MATRICES? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 525
      IF(ANSW.EQ.'Y') GO TO 142
 530  PRINT*,'MORE RUNS WITH NEW MODEL? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 530
      IF(ANSW.EQ.'Y') GO TO 20
      END
C
C END PROGRAM ODEF15 ----------------------------------------
C
C
       SUBROUTINE FSTOL(T,X,DX)
C
C
C
C*********************************************************************
C
```

```
C  THIS IS A SET OF F!    ORDER ORDINARY DIFFERENTIAL EQUATIONS THAT
C  DEFINE THE THE DYNAMICS OF THE STOL F-15 AIRCRAFT.
C  ACTUATOR DYNAMICS ARE INCLUDED AS ENTERED IN THE 12 X 12
C  A MATRIX WHICH HAS BEEN ENTERED AT THE ONSET OF THE PROGRAM
C  IT IS ASSUMED THAT SECOND ORDER ACTUATORS ARE ASSOCIATED WITH
C  THE STABILATOR AND CANARD, AND FIRST ORDER WITH THE NOZZLE.
C  NOTE THAT A NON-LINEARITY IS INTRODUCED INTO THE MODEL BY THE
C  CONTROL OF BOTH THRUST AND NOZZLE DEFLECTION.
C
C**********************************************************************
C
C
      REAL T,X(12),DX(12),BNL(3,3)
      COMMON/MATRIX/UO(2),A(12,12),C(4,12),KX(4,12),KZ(4,4),KXM(4,4)
     1,KXU(4,4),PHI(4,4),PHINT(4,4),CM(4,4),B(3,3),EV(2),KFLAG,MM

      COMMON/CONTRL/UNEW(4),UOLD(4),UCMD(4),UCOLD(4)
     1,XOLD(12),XMOLD(4),XM(4),MFLAG,EVA(2)
C
C
      DO 444 I=1,3
        DO 444 J=1,3
          BNL(I,J)=0.
 444    CONTINUE
C  SET THE SIGN TO ACCOUNT FOR CONTROL SURFACES PASSING
C  THROUGH A ZERO ANGLE OF ATTACK RELATIVE TO THE A/C
C
C
      EVA(1)=EV(1)+X(3)
      EVA(2)=EV(2)+X(3)
      DO 5000 II=1,3
        BNL(II,3)=B(II,3)
          DO 5000 JJ=1,2
            BNL(II,JJ)=B(II,JJ)
      IF(EVA(JJ).GE.0)THEN
        IF((UO(JJ)+EVA(JJ)).LT.0) THEN
          BNL(1,JJ)=-B(1,JJ)
        ENDIF
      ELSE
        IF((UO(JJ)+EVA(JJ)).GT.0) THEN
           BNL(1,JJ)=-B(1,JJ)
      ENDIF
      ENDIF
5000     CONTINUE
      IF(MM.EQ.1)THEN
      DX(1)=A(1,1)*X(1)+A(1,2)*X(2)+A(1,3)*X(3)+A(1,4)*X(4)
     1+BNL(1,1)*UNEW(1)+BNL(1,2)*UNEW(2)+BNL(1,3)*UNEW(3)
      DX(2)=A(2,1)*X(1)+A(2,2)*X(2)+A(2,3)*X(3)+A(2,4)*X(4)
     1+BNL(2,1)*UNEW(1)+BNL(2,2)*UNEW(2)
      DX(3)=A(3,1)*X(1)+A(3,2)*X(2)+A(3,3)*X(3)+A(3,4)*X(4)
     1+BNL(3,1)*UNEW(1)+BNL(3,2)*UNEW(2)
       DX(4)=X(2)
      DX(5)=0.
```

```
      DX(6)=0.
      DX(7)=0.
      DX(8)=0.
      DX(9)=0.
      DX(10)=0.
      DX(11)=0.
      DX(12)=0.
       UO(1)=UNEW(1)
       UO(2)=UNEW(2)
      X(5)=UNEW(1)
       X(7)=UNEW(2)
      X(9)=UNEW(3)
C
      ELSE
        DX(1)=A(1,1)*X(1)+A(1,2)*X(2)+A(1,3)*X(3)+A(1,4)*X(4)
     1+BNL(1,1)*X(5)+BNL(1,2)*X(7)+BNL(1,3)*X(9)
        DX(2)=A(2,1)*X(1)+A(2,2)*X(2)+A(2,3)*X(3)+A(2,4)*X(4)
     1+BNL(2,1)*X(5)+BNL(2,2)*X(7)
        DX(3)=A(3,1)*X(1)+A(3,2)*X(2)+A(3,3)*X(3)+A(3,4)*X(4)
     1+BNL(3,1)*X(5)+BNL(3,2)*X(7)
        DX(4)=X(2)
        DX(5)=X(6)
        DX(6)=-8356.*X(5)-303.*X(6)+8356.*UNEW(1)
        DX(7)=X(8)
        DX(8)=-8356.*X(7)-303.*X(8)+8356.*UNEW(2)
        DX(9)=-20.*X(9)+20.*UNEW(3)
        DX(10)=0.
        DX(11)=0.
        DX(12)=0.
       UO(1)=X(5)
       UO(2)=X(7)
      END IF
C
      IF(NFLAG.EQ.1)THEN
         IF(X(5).GE..262.AND.DX(5).GT.0.0)DX(5)=0.0
         IF(X(5).LE.-.611.AND.DX(5).LT.0.0)DX(5)=0.0
         IF(X(6).GE..401.AND.DX(6).GT.0.0)DX(6)=0.0
         IF(X(6).LE.-.401.AND.DX(6).LT.0.0)DX(6)=0.0
         IF(X(7).GE..262.AND.DX(7).GT.0.0)DX(7)=0.0
         IF(X(7).LE.-.506.AND.DX(7).LT.0.0)DX(7)=0.0
      IF(X(8).GE..803.AND.DX(8).GT.0.0)DX(8)=0.0
         IF(X(8).LE.-.803.AND.DX(8).LT.0.0)DX(8)=0.0
      END IF
       RETURN
       END
C
C END PUBPROGRAM FSTOL----------------------------------------------
C
C
      SUBROUTINE BCSTAR(X,NFLAG)
C
C*********************************************************************
C
```

```
C SUBROUTINE TO CALCULATE THE CONTROLS AT EACH SAMPLE TIME.
C ANTI-WINDUP COMPENSATED IF NFLAG=1.
C
C*********************************************************************
C
      REAL X(12),DEL(12),DEL2(12)
      INTEGER NFLAG
      COMMON/MATRIX/UO(2),A(12,12),C(4,12),KX(4,12),KZ(4,4),KXM(4,4),
     1 KXU(4,4),PHI(4,4),PHINT(4,4),CM(4,4),B(3,3),EV(2),KFLAG,MM
      COMMON/CONTRL/UNEW(4),UOLD(4),UCMD(4),UCOLD(4),XOLD(12),
     1 XMOLD(4),XM(4),NFLAG,EVA(2)
      CALL MATML(PHI,XMOLD,XM,4,4,1)
      CALL MATML(PHINT,UCMD,DEL,4,4,1)
      CALL MATAD(XM,DEL,XM,4,1)
      CALL MATSB(X,XOLD,DEL,12,1)
      CALL MATML(KX,DEL,DEL2,4,12,1)
      CALL MATSB(UOLD,DEL2,UNEW,4,1)
      CALL MATSB(XM,XMOLD,DEL,4,1)
      CALL MATML(KXM,DEL,DEL2,4,4,1)
      CALL MATAD(UNEW,DEL2,UNEW,4,1)
      CALL MATSB(UCMD,UCOLD,DEL,4,1)
      CALL MATML(KXU,DEL,DEL2,4,4,1)
      CALL MATAD(UNEW,DEL2,UNEW,4,1)
      CALL MATML(CM,XMOLD,DEL,4,4,1)
      CALL MATML(C,XOLD,DEL2,4,12,1)
      CALL MATSB(DEL,DEL2,DEL,4,1)
      CALL MATML(KZ,DEL,DEL2,4,4,1)
      CALL MATAD(UNEW,DEL2,UNEW,4,1)
      IF(NFLAG.EQ.1) THEN
      IF(UNEW(1).GT..49-.87*X(5))UNEW(1)=(.49-.87*X(5))*1.
      IF(UNEW(1).LT.-1.14-.87*X(5))UNEW(1)=(-1.14-.87*X(5))*1.
      IF(UNEW(2).GT..49-.87*X(7))UNEW(2)=.49-.87*X(7)
      IF(UNEW(2).LT.-.96-.87*X(7))UNEW(2)=-.96-.87*X(7)
      IF(UNEW(1).GT..706+X(5))UNEW(1)=.706+X(5)
      IF(UNEW(1).LT.-.706+X(5))UNEW(1)=-.706+X(5)
      IF(UNEW(2).GT.1.522+X(7))UNEW(2)=1.522+X(7)
      IF(UNEW(2).LT.-1.522+X(7))UNEW(2)=-1.522+X(7)
      END IF
      RETURN
      END
C
C END SUBROUTINE GCSTAR ----------------------------------------
C
C
      SUBROUTINE RPOUT(A,M,N)
C
C*********************************************************************
C
C THIS ROUTINE PRINTS OUT A REAL MATRIX A
C
C*********************************************************************
C
      REAL A(M,N)
```

```
      INTEGER I,J,N,M
      DO 200 I=1,M
         PRINT'(" ",5(E11.4,3X))',(A(I,J),J=1,N)
         PRINT*
 200  CONTINUE
      END
C
C END SUBROUTINE RPOUT ------------------------------------------
C
C
      SUBROUTINE SETPLT(A,N,M,X)
C
C**********************************************************************
C
C THIS ROUTINE CONVERTS A REAL MATRIX OF DIMENSION N BY M INTO A
C VECTOR THAT IS COMPATIBLE WITH R.M. FLOYD'S PRINTER PLOTTING
C ROUTINE, PLOTLP.  THE INPUT MATRIX IS A.
C N= ROW DIMENSION OF A, THE NUMBER OF POINTS TO BE PLOTTED
C M= COLUMN DIMENSION OF A, THE NUMBER OF FUNCTIONS TO BE PLOTTED +1
C X= THE PLOTTING VECTOR, DIMENSION N*M
C
C**********************************************************************
C
      REAL A(N,M),X(N*M)
      INTEGER N,M,I,J
      DO 100 J=1,M
         DO 100 I=1,N
            X(I+(J-1)*N)=A(I,J)
 100  CONTINUE
      END
C
C END SUBROUTINE SETPLT ------------------------------------------
C
C
      SUBROUTINE PLOTLP(A,N,M,IPSC,ISCL,LPTERM,TITLE)
C
C**********************************************************************
C
C THIS ROUTINE WAS ADAPTED FROM R.M. FLOYD'S THESIS TO PRODUCE
C PRINTER PLOTS OF COMPUTED RESULTS.
C A= VECTOR OF DATA, CONVERTED FROM MATRIX FORM BY SUBROUTINE SETPLT
C N= NUMBER OF POINTS (INDEPENDENT VARIABLE) TO BE PLOTTED
C M= NUMBER OF FUNCTIONS (DEPENDENT VARIABLES) TO BE PLOTTED
C IPSC = -1-->ALL VARIABLES SCALED TOGETHER (1 PLOT)
C      =  0-->SCALED TOGETHER AND SEPARATELY (2 PLOTS)
C      = +1-->SCALED SEPARATELY (1 PLOT)
C ISCL =  0-->PLOT OVER EXACT RANGE OF VARIABLE
C        +1-->PLOT WITH EVEN SCALING
C LPTERM = 0-->PLOT 50 CHARACTERS WIDE
C         +1-->PLOT 100 CHARACTERS WIDE
C TITLE = MAX OF 50 CHARACTERS, TYPE CHARACTER
C
C**********************************************************************
```

```
C
      REAL YSCAL(6),YMIN(6),YPR(11),RISPAC,RMIN,RMAX,YL,YH,XPR,A(*)
      REAL SCAL
      INTEGER IBLNK(6),IPSC,ISCL,LPTERM,IPAPER,ISPAC,IPRTI,ISC,J,IC,IX
      INTEGER IL,JP,ITEMP,M1,M2,M,N,ICO,I
      CHARACTER TITLE*50
      CHARACTER*1 BLANK,PLUS,COLON,GRID,SYMBOL(6),OUT(101)
      DATA BLANK,PLUS,COLON,SYMBOL(1),SYMBOL(2)/' ','+',':','1','2'/
      DATA SYMBOL(3),SYMBOL(4),SYMBOL(5),SYMBOL(6)/'3','4','5','6'/
      IPAPER=5*(1+LPTERM)
      ISPAC=10*IPAPER
      RISPAC=REAL(ISPAC)
      ISPAC=ISPAC+1
      IPRTI=IPAPER+1
      RMIN=A(N+1)
      RMAX=RMIN
25    DO 41 ISC=1,M
         M1=ISC*N+1
         YL=A(M1)
         YH=YL
         M2=N*(ISC+1)
         DO 40 J=M1,M2
            IF(A(J).LT.YL)THEN
               YL=A(J)
            END IF
            IF(A(J).GT.YH)THEN
               YH=A(J)
            END IF
40       CONTINUE
         IF(YL.LT.RMIN)THEN
            RMIN=YL
         END IF
         IF(YH.GT.RMAX)THEN
            RMAX=YH
         END IF
         IF(IPSC.GE.0)THEN
            CALL VARSCL(YL,YH,YSCAL(ISC),RISPAC,ISCL)
         END IF
         YMIN(ISC)=YL
41    CONTINUE
      IF(IPSC.LE.0) THEN
         CALL VARSCL(RMIN,RMAX,SCAL,RISPAC,ISCL)
      END IF
      IC=2-IABS(IPSC)
      DO 42 IX=1,ISPAC
         OUT(IX)=BLANK
42    CONTINUE
      DO 100,ICO=1,IC
         PRINT'("1",11X,A50)',TITLE
         WRITE(4,'(11X,A50)')TITLE
         WRITE(4,'(A1)')BLANK
         PRINT*
         DO 60 I=1,N
```

```
            XPR=A(I)
            IF(MOD(I,10).EQ.0)THEN
               GRID=COLON
            ELSE
               GRID=BLANK
            END IF
            DO 44 IX=2,ISPAC,2
               OUT(IX)=GRID
44          CONTINUE
            DO 46 IX=1,ISPAC,10
               OUT(IX)=PLUS
46          CONTINUE
            DO 55 J=1,M
               IL=I+J*N
               IF(IPSC.EQ.-1)THEN
                 JP=INT((A(IL)-RMIN)/SCAL)+1
               ELSE IF(IPSC.EQ.0)THEN
                  IPS    .PSC+ICO
                  IF(IPSCT.EQ.2)THEN
                     JP=INT((A(IL)-YMIN(J))/YSCAL(J))+1
                  ELSE
                     JP=INT((A(IL)-RMIN)/SCAL)+1
                  END IF
               ELSE
                     JP=INT((A(IL)-YMIN(J))/YSCAL(J))+1
               END IF
50             OUT(JP)=SYMBOL(J)
               IBLNK(J)=JP
55          CONTINUE
            PRINT'(" ",F11.4,6X,101A1)',XPR,(OUT(IX),IX=1,ISPAC)
            WRITE(4,'(F11.4,6X,101A1)')XPR,(OUT(IX),IX=1,ISPAC)
            DO 59 J=1,M
               ITEMP=IBLNK(J)
               OUT(ITEMP)=BLANK
59          CONTINUE
60       CONTINUE
         IF(IPSC.NE.1)THEN
            IF(IPSCT.NE.2)THEN
               YPR(1)=RMIN
               DO 70 I=1,IPAPER
                  YPR(I+1)=YPR(I)+10.*SCAL
70             CONTINUE
               PRINT'("0    SCALE   ",11E10.3)',(YPR(I),I=1,IPRTI)
               WRITE(4,'(A1)')BLANK
               WRITE(4,'("    SCALE   ",11E10.3)')(YPR(I),I=1,IPRTI)
               WRITE(4,'(A1)')BLANK
               WRITE(4,'(A1)')BLANK
            END IF
         END IF
         IF(IPSC.EQ.1.OR.IPSCT.EQ.2)THEN
            DO 76 ISC=1,M
               YPR(1)=YMIN(ISC)
               DO 74 I=1,IPAPER
```

```
                  YPR(I+1)=YPR(I)+10.*YSCAL(ISC)
 74               CONTINUE
                  PRINT'("0    SCALE ",A1,1X,11E10.3)',SYMBOL(ISC),(YPR(IX)
     1,IX=1,IPRTI)
                  WRITE(4,'(A1)')BLANK
                  WRITE(4,'("    SCALE ",A1,1X,11E10.3)')SYMBOL(ISC),
     1(YPR(IX),IX=1,IPRTI)
 76            CONTINUE
            END IF
            DO 90 ISC=1,56-N
               WRITE(4,'(A1)')BLANK
 90         CONTINUE
 100     CONTINUE
         PRINT'("1")'
         END
C
C END SUBROUTINE PLOTLP ------------------------------------------------
C
C
      SUBROUTINE VARSCL(XMIN,XMAX,SCALE,RSPACE,ISCL)
C
C**********************************************************************
C
C THIS IS A SCALING ROUTINE THAT SUPPORTS PLOTLP
C ADAPTED FROM R.M. FLOYD'S THESIS
C
C**********************************************************************
C
      REAL XMIN,XMAX,SCALE,RSPACE,EXP,XMINT,XMAXT
      INTEGER ISCL,ISCAL
      IF(XMAX.EQ.XMIN)THEN
         XMIN=.9*XMIN-10.
      END IF
      SCALE=XMAX-XMIN
      IF(ISCL.NE.0)THEN
         EXP=INT(100.+LOG10(SCALE))-100.
         FACTOR=10.**(1.-EXP)
         XMINT=XMIN*FACTOR
         XMAXT=XMAX*FACTOR
         IF(XMAXT.GE.0.)THEN
            XMAXT=XMAXT+.9
         END IF
         IF(XMINT.LE.0.)THEN
            XMINT=XMINT-.9
         END IF
         XMINT=AINT(XMINT)
         ISCAL=XMAXT-XMINT
         IF(MOD(ISCAL,5).NE.0)THEN
            ISCAL=ISCAL+5-MOD(ISCAL,5)
         END IF
         FACTOR=10.**(EXP-1.)
         XMIN=XMINT*FACTOR
         SCALE=FACTOR*REAL(ISCAL)
```

```
      END IF
      SCALE=SCALE/RSPACE
      END
C
C END SUBROUTINE VARSCL ------------------------------------------
C
C
      SUBROUTINE EDIT(EDMAT,M,N)
C
C*************************************************************************
C
C THIS ROUTINE ALLOWS THE USER TO EDIT AN M BY N MATRIX EDMAT
C
C*************************************************************************
C
      REAL EL,EDMAT(M,N)
      INTEGER M,N,I,J
      CHARACTER ANSW*1
 10   PRINT*,'LIST CURRENT VALUES? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 10
      IF(ANSW.EQ.'Y') CALL RPOUT(EDMAT,M,N)
      PRINT*,'ENTER 0,0,0 <CR> WHEN ALL CHANGES HAVE BEEN MADE'
 100  PRINT*,'ENTER ROW #, COLUMN #, AND MATRIX ELEMENT: '
 110  READ*,I,J,EL
      IF(I.GT.0.AND.I.LE.M.AND.J.GT.0.AND.J.LE.N)THEN
         EDMAT(I,J)=EL
         GO TO 110
      ELSE IF(I.EQ.0.AND.J.EQ.0)THEN
 150  PRINT*,'LIST MODIFIED MATRIX? Y/N: '
      READ(*,'(A)')ANSW
      IF(ANSW.NE.'Y'.AND.ANSW.NE.'N') GO TO 150
      IF(ANSW.EQ.'Y') CALL RPOUT(EDMAT,M,N)
 200     PRINT*,'ANY MORE CHANGES TO THIS MATRIX? Y/N: '
         READ(*,'(A)')ANSW
         IF(ANSW.NE.'Y'.AND.ANSW.NE.'N')GO TO 200
         IF(ANSW.EQ.'Y')GO TO 100
         IF(ANSW.EQ.'N')RETURN
      ELSE
         PRINT*,'SUBSCRIPT OUT OF RANGE'
         GO TO 100
      END IF
      END
C
C END SUBROUTINE EDIT--------------------------------------------
C
C
      SUBROUTINE MATML(A,B,C,L,M,N)
C
C*************************************************************************
C
C THIS ROUTINE WILL MULTIPLY TWO REAL MATRICES
C A=AN L BY M MATRIX
```

```
C B=AN M BY N MATRIX
C C=THE L BY N PRODUCT OF A AND B
C NOTE: ACTUAL ARGUMENT C MUST DIFFER FROM A AND B
C
C*********************************************************************
C
      REAL A(L,M),B(M,N),C(L,N)
      INTEGER I,J,K,L,M,N
      DO 100 I=1,L
         DO 100 J=1,N
            C(I,J)=0.0
 100  CONTINUE
      DO 200 I=1,L
         DO 200 J=1,N
            DO 200 K=1,M
               C(I,J)=C(I,J)+A(I,K)*B(K,J)
 200  CONTINUE
      END
C
C END SUBROUTINE MATML ------------------------------------------
C
C
      SUBROUTINE MATAD(A,B,C,L,M)
C
C*********************************************************************
C
C THIS ROUTINE ADDS TWO REAL MATRICES OF DIMENSION L BY M
C A AND B ARE THE INPUTS, C IS THE SUM
C
C*********************************************************************
C
      REAL A(L,M),B(L,M),C(L,M)
      INTEGER I,J,L,M
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A(I,J)+B(I,J)
 100  CONTINUE
      END
C
C END SUBROUTINE MATAD ------------------------------------------
C
C
      SUBROUTINE MATSB(A,B,C,L,M)
C
C*********************************************************************
C
C THIS ROUTINE SUBTRACTS REAL MATRIX B FROM REAL MATRIX A
C DIFFERENCE IS RETURNED IN REAL MATRIX C.
C ALL THREE MATRICES ARE OF DIMENSION L BY M
C
C*********************************************************************
C
      REAL A(L,M),B(L,M),C(L,M)
```

```
      INTEGER I,J,L,M
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A(I,J)-B(I,J)
  100 CONTINUE
      END
C
C END SUBROUTINE MATSB ------------------------------------------
C
C
      SUBROUTINE SMUL(A,B,C,L,M)
C
C*****************************************************************
C
C THIS ROUTINE MULTIPLIES A REAL MATRIX BY A REAL SCALAR
C A= THE SCALAR
C B= THE MATRIX
C C= THE PRODUCT
C B AND C ARE OF DIMENSION L BY M
C
C*****************************************************************
C
      REAL A,B(L,M),C(L,M)
      INTEGER I,J,L,M
      DO 100 I=1,L
         DO 100 J=1,M
            C(I,J)=A*B(I,J)
  100 CONTINUE
      END
C
C END SUBROUTINE SMUL -------------------------------------------
C
C
      SUBROUTINE COPYMT(A,B,N,M)
C
C*****************************************************************
C
C THIS ROUTINE COPIES A REAL MATRIX A INTO REAL MATRIX B.
C BOTH MATRICES ARE OF DIMENSION N BY M.
C
C*****************************************************************
C
      REAL A(N,M),B(N,M)
      INTEGER I,J,N,M
      DO 100 I=1,N
         DO 100 J=1,M
            B(I,J)=A(I,J)
  100 CONTINUE
      END
C
C END SUBROUTINE COPYMT -----------------------------------------
C
C
```

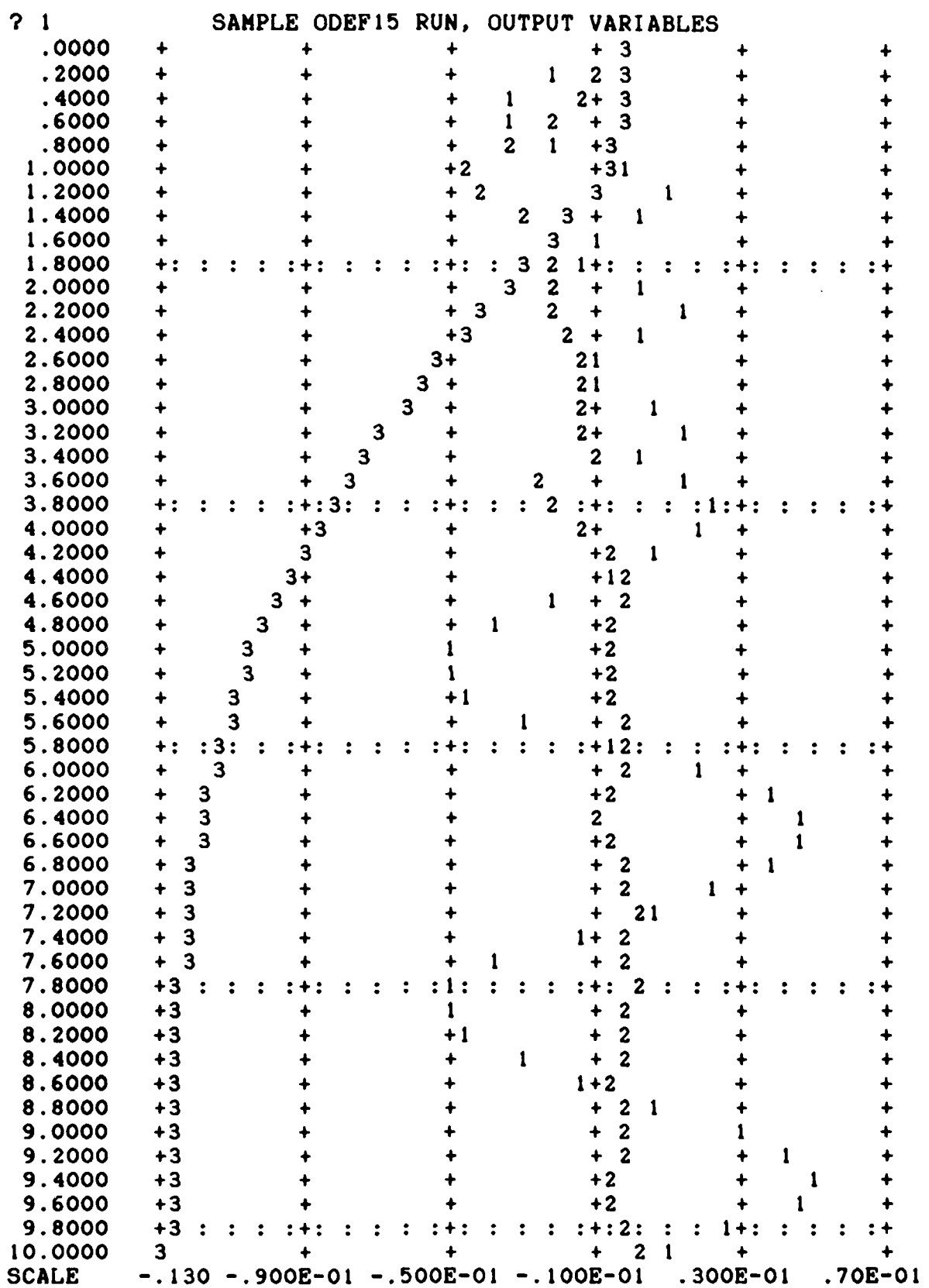
```
      SUBROUTINE DSCRT(A,N,TSAMP,PHI,PHINT,M,TP,TIDENT,CWORK)
C
C*****************************************************************
C
C THIS ROUTINE APPROXIMATES THE STATE TRANSITION MATRIX AND ITS
C INTEGRAL FOR A TIME INVARIANT LINEAR SYSTEM AS A MATRIX EXPONENTIAL
C OVER A SMALL SAMPLE PERIOD.  RESULTS RETURNED IN REAL MATRICES.
C A= SYSTEM DYNAMICS MATRIX, TYPE REAL
C N= STATE DIMENSION
C TSAMP= SAMPLING PERIOD
C PHI= STATE TRANSITION MATRIX, TYPE REAL
C PHINT= APPROXIMATE INTEGRAL OF PHI, TYPE REAL
C M= NUMBER OF TERMS USED IN EXPONENTIAL EXPANSION
C TP, TIDENT AND CWORK ARE DUMMY ARRAYS
C
C*****************************************************************
C
      REAL A(N,N),PHINT(N,N),PHI(N,N),TIDENT(N,N),TP(N,N)
      REAL CWORK(N,N)
      REAL TSAMP,RIJ
      INTEGER I,J,M,N
      DO 200 I=1,N
         DO 100 J=1,N
            TIDENT(I,J)=0.0
 100     CONTINUE
         TIDENT(I,I)=1.0
 200  CONTINUE
      CALL SMUL(TSAMP,TIDENT,PHINT,N,N)
      CALL COPYMT(PHINT,TP,N,N)
      CALL SMUL(TSAMP,A,PHI,N,N)
      DO 300 I=1,M
         CALL MATML(TP,PHI,CWORK,N,N,N)
         CALL COPYMT(CWORK,TP,N,N)
         RIJ=1.0/REAL(I+1)
         CALL SMUL(RIJ,TP,TP,N,N)
         CALL MATAD(PHINT,TP,PHINT,N,N)
 300  CONTINUE
      CALL MATML(A,PHINT,TP,N,N,N)
      CALL MATAD(TIDENT,TP,PHI,N,N)
C
C END SUBROUTINE DSCRT ----------------------------------------
C
      END
       SUBROUTINE KFILT(Z)
C
C*****************************************************************
C
C SUBROUTINE TO INCORPORATE THE KALMAN FILTER INTO THE LOOP FOR
C NON-LINEAR PERFORMANCE ANALYSIS.
C
C*****************************************************************
C
      COMMON/CONTRL/UNEW(4),UOLD(4),UCMD(4),UCOLD(4),XOLD(12),
```

```
     1 XMOLD(4),XM(4),MFLAG,EVA(2),
     1 H(3,4),PHIX(4,4),BD(4,3),QD(4,4),R(3,3),XHP(4),XHM(4),K(4,3)
      REAL AWORK(4,1),BWORK(4,1),CWORK(3,1),DWORK(3,1),EWORK(4,1)
      REAL Z(3)
      CALL MATML(PHIX,XHP,AWORK,4,4,1)
      CALL MATML(BD,UNEW,BWORK,4,3,1)
      CALL MATAD(AWORK,BWORK,XHM,4,1)
      CALL MATML(H,XHM,CWORK,3,4,1)
      CALL MATSB(Z,CWRK,DWORK,3,1)
      CALL MATML(K,DWORK,EWORK,4,3,1)
      CALL MATAD(XHM,EWORK,XHP,4,1)
C
C END SUBROUTINE KFILT-----------------------------------------------
C
      RETURN
      END
```

A-4 Sample Program Execution

The following pages contain a sample run of ODEF15 on the CDC Cyber computer. ODEC is the compiled binary code for ODEF15 and SCHLUS is the data file for the final controller as given in Chapter 5. The output of the computer is in upper case and user responses are in lower case text. No plots are shown in this appendix, but Figures 5-9 through 5-23 are examples of the available plots.

```
/gt,ode,odec,schlus
FILE ODE RETRIEVED
FILE ODEC RETRIEVED
FILE SCHLUS RETRIEVED
/imsl
IMSL5 ATTACHED
/library(imsl5,ode)
LIBRARY(IMSL5,ODE)
/odec
INCORPORATE KALMAN FILTER? Y/N:
? y
DATA TO BE READ FROM FILE? Y/N:
? y
ENTER NAME OF DATA FILE:
? schlus
ANY CHANGES TO MATRICES? Y/N:
? n
WRITE DATA TO OUTPUT FILE? Y/N:
? n
ENTER SAMPLING TIME:
? .025
ENTER CANARD TRIM ANGLE OF ATTACK
? -.0857
ENTER STABILATOR TRIM ANGLE OF ATTACK IN RADIANS
? -.00457
RATE/POSITION LIMITS? Y/N:
? y
ANTI-WINDUP COMPENSATION? Y/N:
? y
EMPLOY ACTUATOR DYNAMICS? Y/N:
? y
ENTER DESIRED RESPONSE DURATION:
? 10
ENTER I AND X(I); 0,0 TO TERMINATE:
? 0 0
SELECT COMMAND INPUT & STEP MAGNITUDE:
? 1,-.087
ENTER SOURCE OF OUTPUT MEASUREMENTS
 1=SYSTEM STATES   2=OUTPUT VARIABLES
? 1
ENTER STATES TO BE MEASURED (3)
? 1,2,4
ENTER # OF ITERATIONS FOR FILTER AVERAGE
? 5
SELECT SEED VALUE FOR RANDOM # GENERATION
? 354
CONTROL BASED ON XHAT+ OR XHAT-?
 XHAT+=1      XHAT-=2
? 1
   1=OUTPUT VARIABLES  2=CONTROL INPUTS
   3=CONTROL DEFLECTIONS  4=SYSTEM STATES
? 1
   +---------------ENTER TITLE FOR PLOT------------+
```


? 1
SAMPLE ODEF15 RUN, OUTPUT VARIABLES
.0000
.2000
.4000
.6000
.8000
1.0000
1.2000
1.4000
1.6000
1.8000
2.0000
2.2000
2.4000
2.6000
2.8000
3.0000
3.2000
3.4000
3.6000
3.8000
4.0000
4.2000
4.4000
4.6000
4.8000
5.0000
5.2000
5.4000
5.6000
5.8000
6.0000
6.2000
6.4000
6.6000
6.8000
7.0000
7.2000
7.4000
7.6000
7.8000
8.0000
8.2000
8.4000
8.6000
8.8000
9.0000
9.2000
9.4000
9.6000
9.8000
10.0000
SCALE -.130 -.900E-01 -.500E-01 -.100E-01 .300E-01 .70E-01

```
1
MORE OUTPUT PLOTS?
? n
INPUT NAME FOR CALCOMP PLOT OF OUTPUT
? out
INPUT NAME FOR CALCOMP PLOT OF CTRL. DEF.
? ctl
CHANGE MATRICES? Y/N:
? n
MORE RUNS WITH NEW MODEL? Y/N:
? n
133.955 CP SECONDS EXECUTION TIME.
```

APPENDIX B. STOLCAT

B-1 Introduction

STOLCAT is a modification of the Conversion And Transformation (CAT) program originally written by Mr. A. Finley Barfield [3]. It is an interactive program that has been modified to allow for the increased dimensionality associated with the additional control surfaces of the STOL F-15. The program converts the raw aerodynamic data, i.e. the nondimensional coefficients, weight and sizing parameters, into a state-space model representation for the aircraft. STOLCAT can be used for longitudinal axis data, lateral directional data separately, or both sets of data simultaneously.

STOLCAT is written in ANSI FORTRAN 5 and is completely self-contained. The user is prompted for all data needed to run the program, including the units and reference frame of expected inputs. The software code listing and a sample program execution follow.

B-2 STOLCAT Source Listing

```
      PROGRAM STOLCAT
      REAL ALPHA,Q,S,C,B,U,DTHETA,W,BIXX,BIYY,BIZZ,
     1BIXZ,DALPHA,DPR,VT,
     2CZA,CZQ,CZU,CZD1,CZD2,CZD3,CZD4,CZD5,CZD6,CZD7,CZD8,
     3CXA,CXQ,CXU,CXD1,CXD2,CXD3,CXD4,CXD5,CXD6,CXD7,CXD8,
     4CMA,CMQ,CMU,CMD1,CMD2,CMD3,CMD4,CMD5,CMD6,CMD7,CMD8,
     5Z1,ZA,ZH,ZQ,ZU,ZD1,ZD2,ZD3,ZD4,ZD5,ZD6,ZD7,ZD8,
     6XA,XH,XQ,XU,XD1,XD2,XD3,XD4,XD5,XD6,XD7,XD8,
     7M1,MA,MH,MQ,MU,MD1,MD2,MD3,MD4,MD5,MD6,MD7,MD8
      REAL CNB,CYB,CLB,L,N
      DIMENSION AMAT(4,4),BMAT(4,8)
      DIMENSION DIRMAT(5,5),DIRBMAT(5,9)
      CHARACTER*3 KEY, KEY1, DATA1, DATA2, DATA3, RUN
      CHARACTER*1 STAB1,STAB2
      DATA Q /48.1/,S /608./, C /15.94/, B /42.7/, U /201./
      DATA DTHETA /11.8030/, DALPHA /11.8030/,W /33576.14/
      DATA BIXX /23644./,BIYY /181847./,BIZZ /199674./,BIXZ /-3086./
      DATA CZA /-7.84976E-2/, CXA /1.5095276E-3/, CMA /9.574118E-3/
      DATA CZQ /0./, CXQ /0./, CMQ /-.16951603/
      DATA CZU /-1.06551597/, CXU /-6.1932E-3/, CMU /6.394289E-2/
      DATA CZH /-1.676463E-4/, CXH /6.662777E-4/, CMH /1.76622E-4/
      DATA CZD1 /-2.63634E-3/, CXD1 /-1.552420E-3/, CMD1 /5.57696E-3/
      DATA CZD2 /-8.31511E-3/, CXD2 /-2.749671E-4/, CMD2 /-1.02066E-2/
      DATA CZD3 /-5.59102E-3/, CXD3 /1.157373E-3/, CMD3 /8.52107E-4/
      DATA CZD4 /-4.50843E-3/, CXD4 /9.4211093E-4/, CMD4 /-2.11118E-3/
      DATA CZD5 /1.896349E-3/, CXD5 /-3.120989E-3/, CMD5 /2.55459E-3/
      DATA CZD6 /-7.422954E-4/, CXD6 /-3.595656E-3/, CMD6 /-1.30123E-3/
      DATA CZD7 /1.896349E-3/, CXD7 /-3.120989E-3/, CMD7 /2.55459E-3/
      DATA CZD8 /-7.422954E-4/, CXD8 /-3.595658E-3/, CMD8 /-1.30123E-3/
      DATA CLB /-2.973933E-3/, CNB /-5.5065055E-4/, CYB /-1.637941E-2/
      DATA CLP /-5.740524E-3/, CNP /-2.3099719E-3/, CYP / 0.000000000/
      DATA CLR / 3.902348E-3/, CNR /-9.6998151E-3/, CYR / 0.000000000/
      DATA CLD1/1.0017E-4/, CND1/-1.3256E-3/, CYD1/3.0606E-3/
      DATA CLD2/-1.14999E-4/, CND2/5.1323E-4/, CYD2/1.3139E-3/
      DATA CLD3/8.5104E-4/, CND3/4.4837E-4/, CYD3/-1.0622E-3/
      DATA CLD4/7.5284E-4/, CND4/7.6138E-5/, CYD4/-1.5235E-4/
      DATA CLD5/6.9959E-4/, CND5/0.00/, CYD5/0.00/
      DATA CLD6/9.6816E-5/, CND6/1.5934E-4/, CYD6/0.0/
      DATA CLD7/-3.7897E-5/, CND7/1.8357E-4/, CYD7/0.0/
      DATA CLD8/-9.6816E-5/, CND8/-1.5934E-4/, CYD8/0.0/
      DATA CLD9/3.7897E-5/, CND9/-1.8357E-4/, CYD9/0.0/
      DPR = 57.2957795
      WRITE(*,5) FORMAT(1X,'*******************************************************')
      WRITE(*,10)
10    FORMAT(1X,'*** STABILITY DERIVATIVE TRANSFORMATION PROGRAM  ***')
      WRITE(*,20)
20    FORMAT(1X,'*******************************************************')
      WRITE(*,100)
100   FORMAT(1X,'ENTER BODY AXIS (NON-DIMENSIONALIZED) COEFFICIENTS ')
      WRITE(*,101)
101   FORMAT(1X,'FOR TRANSFORMATION TO DIMENSIONALIZED BODY AXIS')
      WRITE(*,102)
102   FORMAT(1X,'AND TO GENERATE STATE AND INPUT MATRICES.')
```

```
      WRITE(*,41)
41    FORMAT(1X,'NOTE:  ALL COEFFICIENTS ARE REQUESTED WHEN COMPUTING')
103   CONTINUE
      WRITE(*,30)
30    FORMAT(1X,'*******************************************************')
      WRITE(*,106)
106   FORMAT(1X,'TO TRANSFORM ONLY LONGITUDINAL DATA - TYPE LONG')
      WRITE(*,107)
107   FORMAT(1X,'TO TRANSFORM ONLY LATERAL-DIRECTIONAL DATA - TYPE LAT')
      WRITE(*,108)
108   FORMAT(1X,'TO TRANSFORM BOTH LONG AND LAT-DIR DATA - TYPE BOTH')
      WRITE(*,111)
111   FORMAT(1X,'KEYWORD =   ')
      READ(*,109) KEY
109   FORMAT(A3)
      IF(KEY .EQ. 'LAT') GO TO 104
      IF(KEY .EQ. 'LON') GO TO 104
      IF(KEY .EQ. 'BOT') GO TO 104
      IF(KEY .EQ. 'GAM') GO TO 596
      GO TO 103
104   CONTINUE
      WRITE(*,500)
500   FORMAT(1X,'*******************************************************')
      WRITE(*,510)
510   FORMAT(1X,'Q  (DYNAMIC PRESSURE - LBS/FT**2) = ')
      READ(*,*) Q
      WRITE(*,520)
520   FORMAT(1X,'S  (WING REFERENCE AREA - FT**2) = ')
      READ(*,*) S
      WRITE(*,530)
530   FORMAT(1X,'C  (WING MEAN AERODYNAMIC CORD - FT) = ')
      READ(*,*) C
      WRITE(*,540)
540   FORMAT(1X,'B  (WING SPAN - FT) = ')
      READ(*,*) B
      WRITE(*,550)
550   FORMAT(1X,'VT (TRIM VELOCITY - FT/SEC) = ')
      READ (*,*) U
      VT=U
      WRITE(*,560)
560   FORMAT(1X,'THETA (PITCH ANGLE - DEGS) = ')
      READ(*,*) DTHETA
      WRITE(*,570)
570   FORMAT(1X,'W  (WEIGHT - LBS) = ')
      READ(*,*) W
      WRITE(*,575)
575   FORMAT(1X,'INERTIAS MUST BE INPUT IN BODY AXIS.')
      WRITE(*,580)
580   FORMAT(1X,'IXX  (SLUG-FT**2) = ')
      READ(*,*) BIXX
      WRITE(*,585)
585   FORMAT(1X,'IYY  (SLUG-FT**2) = ')
      READ(*,*) BIYY
```

```
      PROGRAM STOLCAT
      REAL ALPHA,Q,S,C,B,U,DTHETA,W,BIXX,BIYY,BIZZ,
     1BIXZ,DALPHA,DPR,VT,
     2CZA,CZQ,CZU,CZD1,CZD2,CZD3,CZD4,CZD5,CZD6,CZD7,CZD8,
     3CXA,CXQ,CXU,CXD1,CXD2,CXD3,CXD4,CXD5,CXD6,CXD7,CXD8,
     4CMA,CMQ,CMU,CMD1,CMD2,CMD3,CMD4,CMD5,CMD6,CMD7,CMD8,
     5Z1,ZA,ZH,ZQ,ZU,ZD1,ZD2,ZD3,ZD4,ZD5,ZD6,ZD7,ZD8,
     6XA,XH,XQ,XU,XD1,XD2,XD3,XD4,XD5,XD6,XD7,XD8,
     7M1,MA,MH,MQ,MU,MD1,MD2,MD3,MD4,MD5,MD6,MD7,MD8
      REAL CNB,CYB,CLB,L,N
      DIMENSION AMAT(4,4),BMAT(4,8)
      DIMENSION DIRMAT(5,5),DIRBMAT(5,9)
      CHARACTER*3 KEY, KEY1, DATA1, DATA2, DATA3, RUN
      CHARACTER*1 STAB1,STAB2
      DATA Q /48.1/,S /608./, C /15.94/, B /42.7/, U /201./
      DATA DTHETA /11.8030/, DALPHA /11.8030/,W /33576.14/
      DATA BIXX /23644./,BIYY /181847./,BIZZ /199674./,BIXZ /-3086./
      DATA CZA /-7.84976E-2/, CXA /1.5095276E-3/, CMA /9.574118E-3/
      DATA CZQ /0./, CXQ /0./, CMQ /-.16951603/
      DATA CZU /-1.06551597/, CXU /-6.1932E-3/, CMU /6.394289E-2/
      DATA CZH /-1.676463E-4/, CXH /6.662777E-4/, CMH /1.76622E-4/
      DATA CZD1 /-2.63634E-3/, CXD1 /-1.552420E-3/, CMD1 /5.57696E-3/
      DATA CZD2 /-8.31511E-3/, CXD2 /-2.749671E-4/, CMD2 /-1.02066E-2/
      DATA CZD3 /-5.59102E-3/, CXD3 /1.157373E-3/, CMD3 /8.52107E-4/
      DATA CZD4 /-4.50843E-3/, CXD4 /9.4211093E-4/, CMD4 /-2.11118E-3/
      DATA CZD5 /1.896349E-3/, CXD5 /-3.120989E-3/, CMD5 /2.55459E-3/
      DATA CZD6 /-7.422954E-4/, CXD6 /-3.595656E-3/, CMD6 /-1.30123E-3/
      DATA CZD7 /1.896349E-3/, CXD7 /-3.120989E-3/, CMD7 /2.55459E-3/
      DATA CZD8 /-7.422954E-4/, CXD8 /-3.595658E-3/, CMD8 /-1.30123E-3/
      DATA CLB /-2.973933E-3/, CNB /-5.5065055E-4/, CYB /-1.637941E-2/
      DATA CLP /-5.740524E-3/, CNP /-2.3099719E-3/, CYP / 0.000000000/
      DATA CLR / 3.902348E-3/, CNR /-9.6998151E-3/, CYR / 0.000000000/
      DATA CLD1/1.0017E-4/, CND1/-1.3256E-3/, CYD1/3.0606E-3/
      DATA CLD2/-1.14999E-4/, CND2/5.1323E-4/, CYD2/1.3139E-3/
      DATA CLD3/8.5104E-4/, CND3/4.4837E-4/, CYD3/-1.0622E-3/
      DATA CLD4/7.5284E-4/, CND4/7.6138E-5/, CYD4/-1.5235E-4/
      DATA CLD5/6.9959E-4/, CND5/0.00/, CYD5/0.00/
      DATA CLD6/9.6816E-5/, CND6/1.5934E-4/, CYD6/0.0/
      DATA CLD7/-3.7897E-5/, CND7/1.8357E-4/, CYD7/0.0/
      DATA CLD8/-9.6816E-5/, CND8/-1.5934E-4/, CYD8/0.0/
      DATA CLD9/3.7897E-5/, CND9/-1.8357E-4/, CYD9/0.0/
      DPR = 57.2957795
      WRITE(*,5) FORMAT(1X,'*****************************************************')
      WRITE(*,10)
   10 FORMAT(1X,'*** STABILITY DERIVATIVE TRANSFORMATION PROGRAM  ***')
      WRITE(*,20)
   20 FORMAT(1X,'*****************************************************')
      WRITE(*,100)
  100 FORMAT(1X,'ENTER BODY AXIS (NON-DIMENSIONALIZED) COEFFICIENTS ')
      WRITE(*,101)
  101 FORMAT(1X,'FOR TRANSFORMATION TO DIMENSIONALIZED BODY AXIS')
      WRITE(*,102)
  102 FORMAT(1X,'AND TO GENERATE STATE AND INPUT MATRICES.')
```

```
      WRITE(*,590)
590   FORMAT(1X,'IZZ  (SLUG-FT**2) = ')
      READ(*,*) BIZZ
      WRITE(*,595)
595   FORMAT(1X,'IXZ  (SLUG-FT**2) = ')
      READ(*,*) BIXZ
596   CONTINUE
      WRITE(*,597)
597   FORMAT(1X,'**********************************************************')
      WRITE(*,610)
610   FORMAT(16X,'AIRCRAFT PARAMETERS')
      WRITE(*,615) Q
615   FORMAT(1X,'Q  (DYNAMIC PRESSURE - LBS/FT**2) = ',G13.6)
      WRITE(*,620) S
620   FORMAT(1X,'S  (WING REFERENCE AREA - FT**2) = ',G13.6)
      WRITE(*,625) C
625   FORMAT(1X,'C  (WING MEAN AERODYNAMIC CORD - FT) = ',G13.6)
      WRITE(*,630) B
630   FORMAT(1X,'B  (WING SPAN - FT) = ',G13.6)
      WRITE(*,635) U
635   FORMAT(1X,'VT (TRIM VELOCITY - FT/SEC) = ',G13.6)
      WRITE(*,640) DTHETA
640   FORMAT(1X,'THETA = ',G13.6)
      WRITE(*,645) W
645   FORMAT(1X,'W  (WEIGHT - LBS) = ',G13.6)
      WRITE(*,650) BIXX
650   FORMAT(1X,'IXX  (SLUG-FT**2) = ',G13.6)
      WRITE(*,655) BIYY
655   FORMAT(1X,'IYY  (SLUG-FT**2) = ',G13.6)
      WRITE(*,660) BIZZ
660   FORMAT(1X,'IZZ  (SLUG-FT**2) = ',G13.6)
      WRITE(*,665) BIXZ
665   FORMAT(1X,'IXZ  (SLUG-FT**2) = ',G13.6)
      WRITE(*,670)
670   FORMAT(1X,'**********************************************************')
600   CONTINUE
      WRITE(*,675)
675   FORMAT(1X,'IS THE ENTERED DATA CORRECT ?  (YES/NO) ')
      READ(*,680) DATA3
680   FORMAT(A3)
      WRITE(*,685)
685   FORMAT(1X,'**********************************************************')
      IF(DATA3 .EQ. 'NO ') GO TO 104
      IF(DATA3 .EQ. 'YES') GO TO 686
      GO TO 600
686   CONTINUE
      WRITE(*,105)
105   FORMAT(1X,'ALPHA (DEG) = ')
      READ(*,*) DALPHA
      THETA = DTHETA/DPR
      ALPHA = DALPHA/DPR
      IF(KEY .EQ. 'LAT')GO TO 446
      IF(KEY .EQ. 'GAM')GO TO 97
```

```
      WRITE(*,110)
110   FORMAT (1X,'CZA = ')
      READ(*,*) CZA
      WRITE(*,120)
120   FORMAT(1X,'CXA = ')
      READ(*,*) CXA
      WRITE(*,130)
130   FORMAT(1X,'CMA = ')
      READ(*,*) CMA
      WRITE(*,140)
140   FORMAT(1X,'CZQ = ')
      READ(*,*) CZQ
      WRITE(*,150)
150   FORMAT(1X,'CXQ = ')
      READ(*,*) CXQ
      WRITE(*,160)
160   FORMAT(1X,'CMQ = ')
      READ(*,*) CMQ
      WRITE(*,170)
170   FORMAT(1X,'CZU = ')
      READ(*,*) CZU
      WRITE(*,180)
180   FORMAT(1X,'CXU = ')
      READ(*,*) CXU
      WRITE(*,190)
190   FORMAT(1X,'CMU = ')
      READ(*,*) CMU
      WRITE(*,191)
191   FORMAT(1X,'CZH = ')
      READ(*,*) CZH
      WRITE(*,192)
192   FORMAT(1X,'CXH = ')
      READ(*,*) CXH
      WRITE(*,193)
193   FORMAT(1X,'CMH = ')
      READ(*,*) CMH
      WRITE(*,200)
200   FORMAT(1X,'CZD1 = ')
      READ(*,*) CZD1
      WRITE(*,202)
202   FORMAT(1X,'CXD1 = ')
      READ(*,*) CXD1
      WRITE(*,204)
204   FORMAT(1X,'CMD1 = ')
      READ(*,*) CMD1
      WRITE(*,206)
206   FORMAT(1X,'CZD2 = ')
      READ(*,*) CZD2
      WRITE(*,208)
208   FORMAT(1X,'CXD2 = ')
      READ(*,*) CXD2
      WRITE(*,210)
210   FORMAT(1X,'CMD2 = ')
```

```
      READ(*,*) CMD2
      WRITE(*,212)
212   FORMAT(1X,'CZD3 = ')
      READ(*,*) CZD3
      WRITE(*,214)
214   FORMAT(1X,'CXD3 = ')
      READ(*,*) CXD3
      WRITE(*,216)
216   FORMAT(1X,'CMD3 = ')
      READ(*,*) CMD3
      WRITE(*,218)
218   FORMAT(1X,'CZD4 = ')
      READ(*,*) CZD4
      WRITE(*,45)
45    FORMAT(1X,'CXD4 = ')
      READ(*,*) CXD4
      WRITE(*,50)
50    FORMAT(1X,'CMD4 = ')
      READ(*,*) CMD4
      WRITE(*,55)
55    FORMAT(1X,'CZD5 = ')
      READ(*,*) CZD5
      WRITE(*,60)
60    FORMAT(1X,'CXD5 = ')
      READ(*,*) CXD5
      WRITE(*,65)
65    FORMAT(1X,'CMD5 = ')
      READ(*,*) CMD5
      WRITE(*,70)
70    FORMAT(1X,'CZD6 = ')
      READ(*,*) CZD6
      WRITE(*,75)
75    FORMAT(1X,'CXD6 = ')
      READ(*,*) CXD6
      WRITE(*,80)
80    FORMAT(1X,'CMD6 = ')
      READ(*,*) CMD6
      WRITE(*,85)
85    FORMAT(1X,'CZD7')
      READ(*,*) CZD7
      WRITE(*,88)
88    FORMAT(1X,'CXD7')
      READ(*,*) CXD7
      WRITE(*,90)
90    FORMAT(1X,'CMD7 = ')
      READ(*,*) CMD7
      WRITE(*,92)
92    FORMAT(1X,'CZD8 = ')
      READ(*,*) CZD8
      WRITE(*,94)
94    FORMAT(1X,'CXD8 = ')
      READ(*,*) CXD8
      WRITE(*,96)
```

```
 96   FORMAT(1X,'CMD8 = ')
      READ(*,*) CMD8
 97   CONTINUE
      WRITE(*,225)
 225  FORMAT(1X,'*******************************************************')
      WRITE(*,230) DALPHA
 230  FORMAT(15X,'ALPHA =',G13.6)
      WRITE(*,345)
 345  FORMAT(6X,'LONGITUDINAL NON-DIM BODY AXIS COEFFICIENTS(1/DEG)')
C
      CAL = COS(ALPHA)
      SAL = SIN(ALPHA)
      COSSQ = CAL**2
      SINSQ = SAL**2
      COSSIN = CAL*SAL
      CTH = COS(THETA)
      STH = SIN(THETA)
C
      WRITE(*,360) CZA,CMA,CXA
 360  FORMAT(3X,'CZA = ',G13.6,8X,'CMA = ',G13.6,5X,'CXA = ',G13.6
      WRITE(*,390) CZQ,CMQ,CXQ
 390  FORMAT(3X,'CZQ = ',G13.6,8X,'CMQ = ',G13.6,5X,'CXQ = ',G13.6)
      WRITE(*,400) CZH,CMH,CXH
 400  FORMAT(3X,'CZH = ',G13.6,8X,'CMH = ',G13.6,5X,'CXH = ',G13.6)
      WRITE(*,410) CZU,CMU,CXU
 410  FORMAT(3X,'CZU = ',G13.6,8X,'CMU = ',G13.6,5X,'CXU = ',G13.6)
      WRITE(*,370) CZD1,CMD1,CXD1
 370  FORMAT(2X,'CZD1 = ',G13.6,7X,'CMD1 = ',G13.6,4X,'CXD1 = ',G13.6)
      WRITE(*,380) CZD2,CMD2,CXD2
 380  FORMAT(2X,'CZD2 = ',G13.6,7X,'CMD2 = ',G13.6,4X,'CXD2 = ',G13.6)
      WRITE(*,381) CZD3,CMD3,CXD3
 381  FORMAT(2X,'CZD3 = ',G13.6,7X,'CMD3 = ',G13.6,4X,'CXD3 = ',G13.6)
      WRITE(*,382) CZD4,CMD4,CXD4
 382  FORMAT(2X,'CZD4 = ',G13.6,7X,'CMD4 = ',G13.6,4X,'CXD4 = ',G13.6)
      WRITE(*,383) CZD5,CMD5,CXD5
 383  FORMAT(2X,'CZD5 = ',G13.6,7X,'CMD5 = ',G13.6,4X,'CXD5 = ',G13.6)
      WRITE(*,384) CZD6,CMD6,CXD6
 384  FORMAT(2X,'CZD6 = ',G13.6,7X,'CMD6 = ',G13.6,4X,'CXD6 = ',G13.6)
      WRITE(*,385) CZD7,CMD7,CXD7
 385  FORMAT(2X,'CZD7 = ',G13.6,7X,'CMD7 = ',G13.6,4X,'CXD7 = ',G13.6)
      WRITE(*,386) CZD8,CMD8,CXD8
 386  FORMAT(2X,'CZD8 = ',G13.6,7X,'CMD8 = ',G13.6,4X,'CXD8 = ',G13.6)
      WRITE(*,310)
 310  FORMAT(1X,'*******************************************************')
 315  CONTINUE
      WRITE(*,320)
 320  FORMAT(1X,'IS THE ENTERED DATA CORRECT ?   (YES/NO)')
      READ(*,330) DATA1
 330  FORMAT(A3)
      IF(DATA1 .EQ. 'NO ') GO TO 686
      IF(DATA1 .EQ. 'YES') GO TO 340
      GO TO 315
 340  CONTINUE
```

```
      WRITE(*,420)
  420 FORMAT(1X,'**********************************************************')
      Z1 = (Q*S*32.2)/W
      A = C/(2.0*U)
      THETA = DTHETA/DPR
C
      ZA = Z1*CZA*DPR
      ZH = (Z1/U)*CZH
      ZQ = Z1*A*CZQ*DPR
      ZU = 2.*(Z1/U)*CZU
      ZD1 = Z1*CZD1*DPR
      ZD2 = Z1*CZD2*DPR
      ZD3 = Z1*CZD3*DPR
      ZD4 = Z1*CZD4*DPR
      ZD5 = Z1*CZD5*DPR
      ZD6 = Z1*CZD6*DPR
      ZD7 = Z1*CZD7*DPR
      ZD8 = Z1*CZD8*DPR
C
      XA = Z1*CXA*DPR
      XH = (Z1/U)*CXH
      XQ = Z1*A*CXQ*DPR
      XU = 2.*(Z1/U)*CXU
      XD1 = Z1*CXD1*DPR
      XD2 = Z1*CXD2*DPR
      XD3 = Z1*CXD3*DPR
      XD4 = Z1*CXD4*DPR
      XD5 = Z1*CXD5*DPR
      XD6 = Z1*CXD6*DPR
      XD7 = Z1*CXD7*DPR
      XD8 = Z1*CXD8*DPR
C
      M1 = (Q*S*C)/BIYY
C
      MA = M1*CMA*DPR
      MH = (M1/U)*CMH
      MQ = M1*A*CMQ*DPR
      MU = 2.*(M1/U)*CMU
      MD1 = M1*CMD1*DPR
      MD2 = M1*CMD2*DPR
      MD3 = M1*CMD3*DPR
      MD4 = M1*CMD4*DPR
      MD5 = M1*CMD5*DPR
      MD6 = M1*CMD6*DPR
      MD7 = M1*CMD7*DPR
      MD8 = M1*CMD8*DPR
C
      WRITE(*,700)
  700 FORMAT (5X,'LONGITUDINAL AXIS DIMENSIONAL DERIVATIVES')
      WRITE(*,705)
  705 FORMAT (15X,'BODY AXIS (1/RAD)')
      WRITE(*,710) ZA,MA,XA
  710 FORMAT(4X,'ZA = ',E13.6,9X,'MA = ',E13.6,6X,'XA = ',E13.6)
```

```
      WRITE(*,720) ZQ,MQ,XQ
  720 FORMAT(4X,'ZQ = ',G13.6,9X,'MQ = ',G13.6,6X,'XQ = ',G13.6)
      WRITE(*,730) ZH,MH,XH
  730 FORMAT(4X,'ZH = ',G13.6,9X,'MH = ',G13.6,6X,'XH = ',G13.6)
      WRITE(*,740) ZU,MU,XU
  740 FORMAT(4X,'ZU = ',G13.6,9X,'MU = ',G13.6,6X,'XU = ',G13.6)
       WRITE(*,750) ZD1,MD1,XD1
  750 FORMAT(3X,'ZD1 = ',G13.6,8X,'MD1 = ',G13.6,5X,'XD1 = ',G13.6)
      WRITE(*,760) ZD2,MD2,XD2
  760 FORMAT(3X,'ZD2 = ',G13.6,8X,'MD2 = ',G13.6,5X,'XD2 = ',G13.6)
      WRITE(*,770) ZD3,MD3,XD3
  770 FORMAT(3X,'ZD3 = ',G13.6,8X,'MD3 = ',G13.6,5X,'XD3 = ',G13.6)
      WRITE(*,780) ZD4,MD4,XD4
  780 FORMAT(3X,'ZD4 = ',G13.6,8X,'MD4 = ',G13.6,5X,'XD4 = ',G13.6)
      WRITE(*,790) ZD5,MD5,XD5
  790 FORMAT(3X,'ZD5 = ',G13.6,8X,'MD5 = ',G13.6,5X,'XD5 = ',G13.6)
      WRITE(*,800) ZD6,MD6,XD6
  800 FORMAT(3X,'ZD6 = ',G13.6,8X,'MD6 = ',G13.6,5X,'XD6 = ',G13.6)
      WRITE(*,810) ZD7,MD7,XD7
  810 FORMAT(3X,'ZD7 = ',G13.6,8X,'MD7 = ',G13.6,5X,'XD7 = ',G13.6)
      WRITE(*,820) ZD8,MD8,XD8
  820 FORMAT(3X,'ZD8 = ',G13.6,8X,'MD8 = ',G13.6,5X,'XD8 = ',G13.6)
      WRITE(*,830)
  830 FORMAT(1X,'**********************************************************')
C
C     DEVELOPMENT OF STATE MATRICES
C
C     DEVELOPMENT OF THE PLANT MATRIX - A
C
      VT=U
      AMAT(1,1) = XU
      AMAT(1,2) = -VT*SAL
      AMAT(1,3) = XA
      AMAT(1,4) = -32.2*CTH
      AMAT(2,1) = MU
      AMAT(2,2) = MQ
      AMAT(2,3) = MA
      AMAT(2,4) = 0.0
      AMAT(3,1) = ZU/VT
      AMAT(3,2) = CAL
      AMAT(3,3) = ZA/VT
      AMAT(3,4) = -32.2*STH/VT
      AMAT(4,1) = 0.0
      AMAT(4,2) = 1.0
      AMAT(4,3) = 0.0
      AMAT(4,4) = 0.0
C
C
      WRITE(*,*)
      WRITE(*,850)
  850 FORMAT('1',5X,'LONGITUDNAL STATE MATRIX(BODY AXIS)')
      WRITE(*,*)
      WRITE(*,842)
```

```
 842  FORMAT('0',2X,'FOR STATE1=U,STATE2=Q,STATE3=ALPHA,STATE4=THETA')
      WRITE(*,*)
      DO 855 I=1,4
      WRITE(*,860) (AMAT(I,J),J=1,4)
 855  CONTINUE
 860  FORMAT('0',2X,4(G13.6,4X))
      WRITE(*,*)
C
C     NOW WE'LL GET THE INPUT MATRIX - B
C
      BMAT(1,1) = XD1
      BMAT(1,2) = XD2
      BMAT(1,3) = XD3
      BMAT(1,4) = XD4
      BMAT(1,5) = XD5
      BMAT(1,6) = XD6
      BMAT(1,7) = XD7
      BMAT(1,8) = XD8
      BMAT(2,1) = MD1
      BMAT(2,2) = MD2
      BMAT(2,3) = MD3
      BMAT(2,4) = MD4
      BMAT(2,5) = MD5
      BMAT(2,6) = MD6
      BMAT(2,7) = MD7
      BMAT(2,8) = MD8
      BMAT(3,1) = ZD1/VT
      BMAT(3,2) = ZD2/VT
      BMAT(3,3) = ZD3/VT
      BMAT(3,4) = ZD4/VT
      BMAT(3,5) = ZD5/VT
      BMAT(3,6) = ZD6/VT
      BMAT(3,7) = ZD7/VT
      BMAT(3,8) = ZD8/VT
      DO 865 I=1,8
      BMAT(4,I) = 0.0
 865  CONTINUE
C
C     PRINT OUT THE LONG INPUT MATRIX
C
      WRITE(*,*)
      WRITE(*,870)
 870  FORMAT( '0',5X,'LONGITUDNAL INPUT MATRIX')
      WRITE(*,*)
      WRITE(*,868)
 868  FORMAT(2X,'FOR DEL1=CANARD,DEL2=STAB,DEL3=TEF,DEL4=DR AILERON')
      WRITE(*,869)
 869  FORMAT(2X,'    DEL5=RT RV, DEL6=RB RV, DEL7=LT RV, DEL8=LB RV')
      WRITE(*,*)
      WRITE(*,*)
      WRITE(*,871)
 871  FORMAT('0',5X,'ROW1',11X,'ROW2',11X,'ROW3',11X,'ROW4')
      WRITE(*,*)
```

```
      DO 872 I=1,8
      WRITE(*,880) (BMAT(J,I),J=1,4)
 872  CONTINUE
      WRITE(*,*)
 875  CONTINUE
      WRITE(*,873)
 873  FORMAT(1X,' DO YOU WANT STAB AXIS DATA FOR LONG?(Y/N)')
      READ(*,874) STAB1
 874  FORMAT(A1)
      IF ( STAB1 .EQ. 'Y' ) GO TO 877
      IF ( STAB1 .EQ. 'N' ) GO TO 857
      GO TO 875
 877  CONTINUE
C
C*************************************************************
C*(((((((   *
C*(     CONVERT BODY AXIS DATA TO STABILITY AXIS(   *
C*(       (FOR CHECK WITH MCAIR DATA)((   *
C*(((((((   *
C*(((((((   *
C*************************************************************
C
        SMU =( MU*CAL + (MA/U)*SAL*CAL)
        SMH =( ( SMU / MU ) * MH )
        SMA =( MA * COSSQ - MU * U * SAL )
        SMQ = MQ
        SMD1 = MD1
        SMD2 = MD2
        SMD3 = MD3
        SMD4 = MD4
        SMD5 = MD5
        SMD6 = MD6
        SMD7 = MD7
        SMD8 = MD8
C
        SXU=XU*COSSQ+(ZA/U)*SINSQ*CAL+((XA/U)*CAL+ZU)*SAL*CAL
        SXH = (SXU/XU)*XH
        SXA = XA*CAL**3 -U*ZU*SINSQ - (U*XU - ZA*CAL)*CAL*SAL
        SXQ =( XQ*CAL + ZQ*SAL )
        SXD1 = (XD1*CAL + ZD1*SAL)
        SXD2 =( XD2*CAL + ZD2*SAL)
        SXD3 = (XD3*CAL + ZD3*SAL)
        SXD4 = (XD4*CAL + ZD4*SAL)
        SXD5 = (XD5*CAL + ZD5*SAL)
        SXD6 = (XD6*CAL + ZD6*SAL)
        SXD7 = (XD7*CAL + ZD7*SAL)
        SXD8 = (XD8*CAL + ZD8*SAL)
C
        SZU=ZU*COSSQ-(XA/U)*SINSQ*CAL-(XU-(ZA/U)*CAL)*SAL*CAL
        SZH = (SZU/ZU) * ZH
        SZA=ZA*CAL**3 + U*XU*SINSQ - (U*ZU + XA*CAL)*CAL*SAL
        SZQ = (ZQ*CAL - XQ*SAL)
        SZD1 = (ZD1*CAL - XD1*SAL)
```

```
      SZD2 = (ZD2*CAL - XD2*SAL)
      SZD3 = (ZD3*CAL - XD3*SAL)
      SZD4 = (ZD4*CAL - XD4*SAL)
      SZD5 = (ZD5*CAL - XD5*SAL)
      SZD6 = (ZD6*CAL - XD6*SAL)
      SZD7 = (ZD7*CAL - XD7*SAL)
      SZD8 = (ZD8*CAL - XD8*SAL)
      WRITE(*,701)
  701 FORMAT ('0',5X,'LONGITUDINAL AXIS DIMENSIONAL DERIVATIVES')
      WRITE(*,702)
  702 FORMAT (15X,' STABILITY AXIS (1/RAD) ')
      WRITE(*,711) SZA,SMA,SXA
  711 FORMAT(4X,'ZA = ',G13.6,9X,'MA = ',G13.6,6X,'XA = ',G13.6)
      WRITE(*,721) SZQ,SMQ,SXQ
  721 FORMAT(4X,'ZQ = ',G13.6,9X,'MQ = ',G13.6,6X,'XQ = ',G13.6)
      WRITE(*,731) SZH,SMH,SXH
  731 FORMAT(4X,'ZH = ',G13.6,9X,'MH = ',G13.6,6X,'XH = ',G13.6)
      WRITE(*,741) SZU,SMU,SXU
  741 FORMAT(4X,'ZU = ',G13.6,9X,'MU = ',G13.6,6X,'XU = ',G13.6)
      WRITE(*,751) SZD1,SMD1,SXD1
  751 FORMAT(3X,'ZD1 = ',G13.6,8X,'MD1 = ',G13.6,5X,'XD1 = ',G13.6)
      WRITE(*,761) SZD2,SMD2,SXD2
  761 FORMAT(3X,'ZD2 = ',G13.6,8X,'MD2 = ',G13.6,5X,'XD2 = ',G13.6)
      WRITE(*,771) SZD3,SMD3,SXD3
  771 FORMAT(3X,'ZD3 = ',G13.6,8X,'MD3 = ',G13.6,5X,'XD3 = ',G13.6)
      WRITE(*,781) SZD4,SMD4,SXD4
  781 FORMAT(3X,'ZD4 = ',G13.6,8X,'MD4 = ',G13.6,5X,'XD4 = ',G13.6)
      WRITE(*,791) SZD5,SMD5,SXD5
  791 FORMAT(3X,'ZD5 = ',G13.6,8X,'MD5 = ',G13.6,5X,'XD5 = ',G13.6)
      WRITE(*,800) SZD6,SMD6,SXD6
  801 FORMAT(3X,'ZD6 = ',G13.6,8X,'MD6 = ',G13.6,5X,'XD6 = ',G13.6)
      WRITE(*,811) SZD7,SMD7,SXD7
  811 FORMAT(3X,'ZD7 = ',G13.6,8X,'MD7 = ',G13.6,5X,'XD7 = ',G13.6)
      WRITE(*,820) SZD8,SMD8,SXD8
  821 FORMAT(3X,'ZD8 = ',G13.6,8X,'MD8 = ',G13.6,5X,'XD8 = ',G13.6)
      WRITE(*,830)
  880 FORMAT(2X,4(G13.6,2X))
C
      AMAT(1,1) = SXU
      AMAT(1,2) = 0.0
      AMAT(1,3) = SXA
      AMAT(1,4) = -32.2*CTH
      AMAT(2,1) = SMU
      AMAT(2,2) = SMQ
      AMAT(2,3) = SMA
      AMAT(2,4) = 0.0
      AMAT(3,1) = SZU/U
      AMAT(3,2) = 1.0
      AMAT(3,3) = SZA/U
      AMAT(3,4) = -32.2*STH/U
      AMAT(4,1) = 0.0
      AMAT(4,2) = 1.0
      AMAT(4,3) = 0.0
```

```
       AMAT(4,4) = 0.0
C      WRITE(*,851)
C851   FORMAT('0',5X,'LONGITUDNAL STATE MATRIX (STAB AXIS)')
C      WRITE(*,*)
C      WRITE(*,842)
C      WRITE(*,*)
C      DO 856 I=1,4
C      WRITE(*,860) (AMAT(I,J),J=1,4)
C856   CONTINUE
 857   CONTINUE
       IF (KEY .EQ. 'BOT' ) GO TO 446
       IF (KEY .EQ. 'GAM' ) GO TO 1465
 421   CONTINUE
       WRITE(*,430)
 430   FORMAT(1X,'IS ANOTHER PROGRAM RUN DESIRED ?  (YES/NO)')
       READ(*,440) RUN
 440   FORMAT(A3)
       WRITE(*,445)
 445   FORMAT(1X,'**********************************************************')
       IF(RUN .EQ. 'NO ') GO TO 450
       IF(RUN .EQ. 'YES') GO TO 103
       GO TO 421
 446   CONTINUE
C
C      THIS IS WHERE THE LATERAL DIRECTIONAL STARTS
C
       WRITE(*,1110)
 1110  FORMAT(1X,'CLB (1/DEG) = ')
       READ(*,*) CLB
       WRITE(*,1120)
 1120  FORMAT(1X,'CNB (1/DEG) = ')
       READ(*,*) CNB
       WRITE(*,1130)
 1130  FORMAT(1X,'CYB (1/DEG) = ')
       READ(*,*) CYB
       WRITE(*,1140)
 1140  FORMAT(1X,'CLP (1/DEG) = ')
       READ(*,*) CLP
       WRITE(*,1150)
 1150  FORMAT(1X,'CNP (1/DEG) = ')
       READ(*,*) CNP
       WRITE(*,1160)
 1160  FORMAT(1X,'CYP (1/DEG) = ')
       READ(*,*) CYP
       WRITE(*,1170)
 1170  FORMAT(1X,'CLR (1/DEG) = ')
       READ(*,*) CLR
       WRITE(*,1180)
 1180  FORMAT(1X,'CNR (1/DEG) = ')
       READ(*,*) CNR
       WRITE(*,1190)
 1190  FORMAT(1X,'CYR (1/DEG) = ')
       READ(*,*) CYR
```

```
      WRITE(*,1200)
1200  FORMAT(1X,'CLD1 (1/DEG) = ')
      READ(*,*) CLD1
      WRITE(*,1210)
1210  FORMAT(1X,'CND1 (1/DEG) = ')
      READ(*,*) CND1
      WRITE(*,1220)
1220  FORMAT(1X,'CYD1 (1/DEG) = ')
      READ(*,*) CYD1
      WRITE(*,1230)
1230  FORMAT(1X,'CLD2 (1/DEG) = ')
      READ(*,*) CLD2
      WRITE(*,1240)
1240  FORMAT(1X,'CND2 (1/DEG) = ')
      READ(*,*) CND2
      WRITE(*,1250)
1250  FORMAT(1X,'CYD2 (1/DEG) = ')
      READ(*,*) CYD2
      WRITE(*,1260)
1260  FORMAT(1X,'CLD3 (1/DEG) = ')
      READ(*,*) CLD3
      WRITE(*,1270)
1270  FORMAT(1X,'CND3 (1/DEG) = ')
      READ(*,*) CND3
      WRITE(*,1280)
1280  FORMAT(1X,'CYD3 (1/DEG) = ')
      READ(*,*) CYD3
      WRITE(*,1290)
1290  FORMAT(1X,'CLD4 (1/DEG) = ')
      READ(*,*) CLD4
      WRITE(*,1300)
1300  FORMAT(1X,'CND4 (1/DEG) = ')
      READ(*,*) CND4
      WRITE(*,1310)
1310  FORMAT(1X,'CYD4 (1/DEG) = ')
      READ(*,*) CYD4
      WRITE(*,1320)
1320  FORMAT(1X,'CLD5 (1/DEG) = ')
      READ(*,*) CLD5
      WRITE(*,1330)
1330  FORMAT(1X,'CND5 (1/DEG) = ')
      READ(*,*) CND5
      WRITE(*,1340)
1340  FORMAT(1X,'CYD5 (1/DEG) = ')
      READ(*,*) CYD5
      WRITE(*,1350)
1350  FORMAT(1X,'CLD6 (1/DEG) = ')
      READ(*,*) CLD6
      WRITE(*,1360)
1360  FORMAT(1X,'CND6 (1/DEG) = ')
      READ(*,*) CND6
      WRITE(*,1370)
1370  FORMAT(1X,'CYD6 (1/DEG) = ')
```

```
      READ(*,*) CYD6
      WRITE(*,1380)
 1380 FORMAT(1X,'CLD7 (1/DEG) = ')
      READ(*,*) CLD7
      WRITE(*,1390)
 1390 FORMAT(1X,'CND7 (1/DEG) = ')
      READ(*,*) CND7
      WRITE(*,1400)
 1400 FORMAT(1X,'CYD7 (1/DEG) = ')
      READ(*,*) CYD7
      WRITE(*,1410)
 1410 FORMAT(1X,'CLD8 (1/DEG) = ')
      READ(*,*) CLD8
      WRITE(*,1420)
 1420 FORMAT(1X,'CND8 (1/DEG) = ')
      READ(*,*) CND8
      WRITE(*,1430)
 1430 FORMAT(1X,'CYD8 (1/DEG) = ')
      READ(*,*) CYD8
      WRITE(*,1440)
 1440 FORMAT(1X,'CLD9 (1/DEG) = ')
      READ(*,*) CLD9
      WRITE(*,1450)
 1450 FORMAT(1X,'CND9 (1/DEG) = ')
      READ(*,*) CND9
      WRITE(*,1460)
 1460 FORMAT(1X,'CYD9 (1/DEG) = ')
      READ(*,*) CYD9
 1465 CONTINUE
      WRITE(*,1470)
 1470 FORMAT('1',8X,'LAT-DIR BODY AXIS COEFFICIENTS')
      IF(KEY .EQ. 'LON') GO TO 1490
      IF(KEY .EQ. 'BOT') GO TO 1490
      WRITE(*,1480) DALPHA
 1480 FORMAT(15X,'ALPHA = ',G13.6)
 1490 CONTINUE
      WRITE(*,1500) CLB,CNB,CYB
 1500 FORMAT(3X,'CLB = ',G13.6,8X,'CNB = ',G13.6,5X,'CYB = ',G13.6)
      WRITE(*,1510) CLP,CNP,CYP
 1510 FORMAT(3X,'CLP = ',G13.6,8X,'CNP = ',G13.6,5X,'CYP = ',G13.6)
      WRITE(*,1520) CLR,CNR,CYR
 1520 FORMAT(3X,'CLR = ',G13.6,8X,'CNR = ',G13.6,5X,'CYR = ',G13.6)
      WRITE(*,1530) CLD1,CND1,CYD1
 1530 FORMAT(2X,'CLD1 = ',G13.6,7X,'CND1 = ',G13.6,4X,'CYD1 = ',G13.6)
      WRITE(*,1540) CLD2,CND2,CYD2
 1540 FORMAT(2X,'CLD2 = ',G13.6,7X,'CND2 = ',G13.6,4X,'CYD2 = ',G13.6)
      WRITE(*,1550) CLD3,CND3,CYD3
 1550 FORMAT(2X,'CLD3 = ',G13.6,7X,'CND3 = ',G13.6,4X,'CYD3 = ',G13.6)
      WRITE(*,1560) CLD4,CND4,CYD4
 1560 FORMAT(2X,'CLD4 = ',G13.6,7X,'CND4 = ',G13.6,4X,'CYD4 = ',G13.6)
      WRITE(*,1570) CLD5,CND5,CYD5
 1570 FORMAT(2X,'CLD5 = ',G13.6,7X,'CND5 = ',G13.6,4X,'CYD5 = ',G13.6)
      WRITE(*,1580) CLD6,CND6,CYD6
```

```
1580 FORMAT(2X,'CLD6 = ',G13.6,7X,'CND6 = ',G13.6,4X,'CYD6 = ',G13.6)
     WRITE(*,1590) CLD7,CND7,CYD7
1590 FORMAT(2X,'CLD7 = ',G13.6,7X,'CND7 = ',G13.6,4X,'CYD7 = ',G13.6)
     WRITE(*,1600) CLD8,CND8,CYD8
1600 FORMAT(2X,'CLD8 = ',G13.6,7X,'CND8 = ',G13.6,4X,'CYD8 = ',G13.6)
     WRITE(*,1610) CLD9,CND9,CYD9
1610 FORMAT(2X,'CLD9 = ',G13.6,7X,'CND9 = ',G13.6,4X,'CYD9 = ',G13.6)
     WRITE(*,*)
     WRITE(*,1620)
1620 FORMAT(1X,'**************************************************')
1625 CONTINUE
     WRITE(*,1630)
1630 FORMAT(1X,'IS THE ENTERED DATA CORRECT ?   (YES/NO)')
     READ(*,1640) DATA2
1640 FORMAT(A3)
     IF ( DATA2 .EQ. 'NO') GO TO 446
     IF ( DATA2 .EQ. 'YES' ) GO TO 1645
     GO TO 1625
1645 CONTINUE
     WRITE(*,1646)
1646 FORMAT(1X,'DO YOU WANT STAB AXIS DATA FOR LAT-DIR? (Y/N)')
     READ(*,1647) STAB2
1647 FORMAT(A1)
     IF ( STAB2 .EQ. 'N') GO TO 1801
     IF ( STAB2 .EQ. 'Y') GO TO 1648
     GO TO 1645
1648 CONTINUE
     BSALPH=-ALPHA
     CSA=COS(BSALPH)
     SSA=SIN(BSALPH)
     CS=CSA*CSA
     SS=SSA*SSA
C
     SCLP=CLP*CS + CNR*SS - (CLR + CNP)*CSA*SSA
     SCLR=CLR*CS - CNP*SS + (CLP - CNR)*CSA*SSA
     SCLB=CLB*CSA - CNB*SSA
     SCLD1=CLD1*CSA - CND1*SSA
     SCLD2=CLD2*CSA - CND2*SSA
     SCLD3=CLD3*CSA - CND3*SSA
     SCLD4=CLD4*CSA - CND4*SSA
     SCLD5=CLD5*CSA - CND5*SSA
     SCLD6=CLD6*CSA - CND6*SSA
     SCLD7=CLD7*CSA - CND7*SSA
     SCLD8=CLD8*CSA - CND8*SSA
     SCLD9=CLD9*CSA - CND9*SSA
C
     SCNP=CNP*CS - CLR*SS + (CLP - CNR)*CSA*SSA
     SCNR=CNR*CS + CLP*SS + (CLR + CNP)*CSA*SSA
     SCNB=CNB*CSA + CLB*SSA
     SCND1=CND1*CSA + CLD1*SSA
     SCND2=CND2*CSA + CLD2*SSA
     SCND3=CND3*CSA + CLD3*SSA
     SCND4=CND4*CSA + CLD4*SSA
```

```
      SCND5=CND5*CSA + CLD5*SSA
      SCND6=CND6*CSA + CLD6*SSA
      SCND7=CND7*CSA + CLD7*SSA
      SCND8=CND8*CSA + CLD8*SSA
      SCND9=CND9*CSA + CLD9*SSA
C
      SCYP=CYP*CSA - CYR*SSA
      SCYR=CYR*CSA + CYP*SSA
      SCYB=CYB
      WRITE(*,1471)
 1471 FORMAT(8X,'LAT-DIR STAB AXIS COEFFICIENTS')
      WRITE(*,1501) SCLB,SCNB,SCYB
 1501 FORMAT(3X,'CLB = ',G13.6,8X,'CNB = ',G13.6,5X,'CYB = ',G13.6)
      WRITE(*,1511) SCLP,SCNP,SCYP
 1511 FORMAT(3X,'CLP = ',G13.6,8X,'CNP = ',G13.6,5X,'CYP = ',G13.6)
      WRITE(*,1521) SCLR,SCNR,SCYR
 1521 FORMAT(3X,'CLR = ',G13.6,8X,'CNR = ',G13.6,5X,'CYR = ',G13.6)
      WRITE(*,1531) SCLD1,SCND1,CYD1
 1531 FORMAT(2X,'CLD1 = ',G13.6,7X,'CND1 = ',G13.6,4X,'CYD1 = ',G13.6)
      WRITE(*,1541) SCLD2,SCND2,CYD2
 1541 FORMAT(2X,'CLD2 = ',G13.6,7X,'CND2 = ',G13.6,4X,'CYD2 = ',G13.6)
      WRITE(*,1551) SCLD3,SCND3,CYD3
 1551 FORMAT(2X,'CLD3 = ',G13.6,7X,'CND3 = ',G13.6,4X,'CYD3 = ',G13.6)
      WRITE(*,1561) SCLD4,SCND4,CYD4
 1561 FORMAT(2X,'CLD4 = ',G13.6,7X,'CND4 = ',G13.6,4X,'CYD4 = ',G13.6)
      WRITE(*,1571) SCLD5,SCND5,CYD5
 1571 FORMAT(2X,'CLD5 = ',G13.6,7X,'CND5 = ',G13.6,4X,'CYD5 = ',G13.6)
      WRITE(*,1581) SCLD6,SCND6,CYD6
 1581 FORMAT(2X,'CLD6 = ',G13.6,7X,'CND6 = ',G13.6,4X,'CYD6 = ',G13.6)
      WRITE(*,1591) SCLD7,SCND7,CYD7
 1591 FORMAT(2X,'CLD7 = ',G13.6,7X,'CND7 = ',G13.6,4X,'CYD7 = ',G13.6)
      WRITE(*,1601) SCLD8,SCND8,CYD8
 1601 FORMAT(2X,'CLD8 = ',G13.6,7X,'CND8 = ',G13.6,4X,'CYD8 = ',G13.6)
      WRITE(*,1611) SCLD9,SCND9,CYD9
 1611 FORMAT(2X,'CLD9 = ',G13.6,7X,'CND9 = ',G13.6,4X,'CYD9 = ',G13.6)
      WRITE(*,*)
C
      SIXX=BIXX*COSSQ + BIZZ*SINSQ - BIXZ*SIN(2*ALPHA)
      SIYY=BIYY
      SIZZ=BIZZ*COSSQ + BIXX*SINSQ + BIXZ*SIN(2*ALPHA)
      SIXZ=BIXZ*COS(2*ALPHA) + .5*(BIXX - BIZZ)*SIN(2*ALPHA)
C
      SN = DPR*(Q*S*B)/SIZZ
      SL = DPR*(Q*S*B)/SIXX
      SB = B/(2.0*U)
      SY = DPR*(Q*S*32.2)/W
      SNB = SN*SCNB
      SNP = SN*SB*SCNP
      SNR = SN*SB*SCNR
      SND1 = SN*SCND1
      SND2 = SN*SCND2
      SND3 = SN*SCND3
      SND4 = SN*SCND4
```

```
      SND5 = SN*SCND5
      SND6 = SN*SCND6
      SND7 = SN*SCND7
      SND8 = SN*SCND8
      SND9 = SN*SCND9
C
      SLB = SL*SCLB
      SLP = SL*SB*SCLP
      SLR = SL*SB*SCLR
      SLD1 = SL*SCLD1
      SLD2 = SL*SCLD2
      SLD3 = SL*SCLD3
      SLD4 = SL*SCLD4
      SLD5 = SL*SCLD5
      SLD6 = SL*SCLD6
      SLD7 = SL*SCLD7
      SLD8 = SL*SCLD8
      SLD9 = SL*SCLD9
C
      SYB = SY*SCYB
      SYR = SY*SB*SCYR
      SYP = SY*SB*SCYP
      SYD1 = SY* CYD1
      SYD2 = SY* CYD2
      SYD3 = SY* CYD3
      SYD4 = SY* CYD4
      SYD5 = SY* CYD5
      SYD6 = SY* CYD6
      SYD7 = SY* CYD7
      SYD8 = SY* CYD8
      SYD9 = SY* CYD9
C
      WRITE(*,1661)
 1661 FORMAT(5X,'LAT-DIR STAB AXIS DIMENSIONAL DERIVATIVES(1/RAD)')
      WRITE(*,1671) SNB,SLB,SYB
 1671 FORMAT(4X,'NB = ',G13.6,9X,'LB = ',G13.6,5X,'YB = ',G13.6)
      WRITE(*,1681) SNP,SLP,SYP
 1681 FORMAT(4X,'NP = ',G13.6,9X,'LP = ',G13.6,5X,'YP = ',G13.6)
      WRITE(*,1691) SNR,SLR,SYR
 1691 FORMAT(4X,'NR = ',G13.6,9X,'LR = ',G13.6,5X,'YR = ',G13.6)
      WRITE(*,1701) SND1,SLD1,SYD1
 1701 FORMAT(3X,'ND1 = ',G13.6,8X,'LD1 = ',G13.6,4X,'YD1 = ',G13.6)
      WRITE(*,1711) SND2,SLD2,SYD2
 1711 FORMAT(3X,'ND2 = ',G13.6,8X,'LD2 = ',G13.6,4X,'YD2 = ',G13.6)
      WRITE(*,1721) SND3,SLD3,SYD3
 1721 FORMAT(3X,'ND3 = ',G13.6,8X,'LD3 = ',G13.6,4X,'YD3 = ',G13.6)
      WRITE(*,1731) SND4,SLD4,SYD4
 1731 FORMAT(3X,'ND4 = ',G13.6,8X,'LD4 = ',G13.6,4X,'YD4 = ',G13.6)
      WRITE(*,1741) SND5,SLD5,SYD5
 1741 FORMAT(3X,'ND5 = ',G13.6,8X,'LD5 = ',G13.6,4X,'YD5 = ',G13.6)
      WRITE(*,1751) SND6,SLD6,SYD6
 1751 FORMAT(3X,'ND6 = ',G13.6,8X,'LD6 = ',G13.6,4X,'YD6 = ',G13.6)
      WRITE(*,1761) SND7,SLD7,SYD7
```

```
 1761 FORMAT(3X,'ND7 = ',G13.6,8X,'LD7 = ',G13.6,4X,'YD7 = ',G13.6)
      WRITE(*,1771) SND8,SLD8,SYD8
 1771 FORMAT(3X,'ND8 = ',G13.6,8X,'LD8 = ',G13.6,4X,'YD8 = ',G13.6)
      WRITE(*,1781) SND9,SLD9,SYD9
 1781 FORMAT(3X,'ND9 = ',G13.6,8X,'LD9 = ',G13.6,4X,'YD9 = ',G13.6)
C
      WRITE(*,1650)
 1650 FORMAT(1X,'*******************************************************')
 1801 CONTINUE
      N = DPR*(Q*S*B)/BIZZ
      L = DPR*(Q*S*B)/BIXX
      BB = B/(2.0*U)
      Y = DPR*(Q*S*32.2)/W
      BNB = N*CNB
      BNP = N*BB*CNP
      BNR = N*BB*CNR
      BND1 = N*CND1
      BND2 = N*CND2
      BND3 = N*CND3
      BND4 = N*CND4
      BND5 = N*CND5
      BND6 = N*CND6
      BND7 = N*CND7
      BND8 = N*CND8
      BND9 = N*CND9
C
      BLB = L*CLB
      BLP = L*BB*CLP
      BLR = L*BB*CLR
      BLD1 = L*CLD1
      BLD2 = L*CLD2
      BLD3 = L*CLD3
      BLD4 = L*CLD4
      BLD5 = L*CLD5
      BLD6 = L*CLD6
      BLD7 = L*CLD7
      BLD8 = L*CLD8
      BLD9 = L*CLD9
C
      BYB = Y*CYB
      BYR = Y*BB*CYR
      BYP = Y*BB*CYP
      BYD1 = Y*CYD1
      BYD2 = Y*CYD2
      BYD3 = Y*CYD3
      BYD4 = Y*CYD4
      BYD5 = Y*CYD5
      BYD6 = Y*CYD6
      BYD7 = Y*CYD7
      BYD8 = Y*CYD8
      BYD9 = Y*CYD9
C
      WRITE(*,1660)
```

```
 1660 FORMAT(5X,'LAT-DIR BODY AXIS DIMENSIONAL DERIVATIVES(1/RAD)')
      WRITE(*,1670) BNB,BLB,BYB
 1670 FORMAT(4X,'NB = ',G13.6,9X,'LB = ',G13.6,5X,'YB = ',G13.6)
      WRITE(*,1680) BNP,BLP,BYP
 1680 FORMAT(4X,'NP = ',G13.6,9X,'LP = ',G13.6,5X,'YP = ',G13.6)
      WRITE(*,1690) BNR,BLR,BYR
 1690 FORMAT(4X,'NR = ',G13.6,9X,'LR = ',G13.6,5X,'YR = ',G13.6)
      WRITE(*,1700) BND1,BLD1,BYD1
 1700 FORMAT(3X,'ND1 = ',G13.6,8X,'LD1 = ',G13.6,4X,'YD1 = ',G13.6)
      WRITE(*,1710) BND2,BLD2,BYD2
 1710 FORMAT(3X,'ND2 = ',G13.6,8X,'LD2 = ',G13.6,4X,'YD2 = ',G13.6)
      WRITE(*,1720) BND3,BLD3,BYD3
 1720 FORMAT(3X,'ND3 = ',G13.6,8X,'LD3 = ',G13.6,4X,'YD3 = ',G13.6)
      WRITE(*,1730) BND4,BLD4,BYD4
 1730 FORMAT(3X,'ND4 = ',G13.6,8X,'LD4 = ',G13.6,4X,'YD4 = ',G13.6)
      WRITE(*,1740) BND5,BLD5,BYD5
 1740 FORMAT(3X,'ND5 = ',G13.6,8X,'LD5 = ',G13.6,4X,'YD5 = ',G13.6)
      WRITE(*,1750) BND6,BLD6,BYD6
 1750 FORMAT(3X,'ND6 = ',G13.6,8X,'LD6 = ',G13.6,4X,'YD6 = ',G13.6)
      WRITE(*,1760) BND7,BLD7,BYD7
 1760 FORMAT(3X,'ND7 = ',G13.6,8X,'LD7 = ',G13.6,4X,'YD7 = ',G13.6)
      WRITE(*,1770) BND8,BLD8,BYD8
 1770 FORMAT(3X,'ND8 = ',G13.6,8X,'LD8 = ',G13.6,4X,'YD8 = ',G13.6)
      WRITE(*,1780) BND9,BLD9,BYD9
 1780 FORMAT(3X,'ND9 = ',G13.6,8X,'LD9 = ',G13.6,4X,'YD9 = ',G13.6)
      WRITE(*,1790)
 1790 FORMAT(1X,'**********************************************************')
      WRITE(*,1800)
 1800 FORMAT(1X,'**********************************************************')
C
C
C     CONVERSION OF DATA INTO STATE SPACE FORM
C
C
      D = 1.0 - ((BIXZ*BIXZ)/(BIXX*BIZZ))
      R1 = BIXZ/BIZZ
      R2 = BIXZ/BIXX
C
      PBNB = (BNB + R1*BLB)/D
      PBNP = (BNP + R1*BLP)/D
      PBNR = (BNR + R1*BLR)/D
      PBND1 = (BND1 + R1*BLD1)/D
      PBND2 = (BND2 + R1*BLD2)/D
      PBND3 = (BND3 + R1*BLD3)/D
      PBND4 = (BND4 + R1*BLD4)/D
      PBND5 = (BND5 + R1*BLD5)/D
      PBND6 = (BND6 + R1*BLD6)/D
      PBND7 = (BND7 + R1*BLD7)/D
      PBND8 = (BND8 + R1*BLD8)/D
      PBND9 = (BND9 + R1*BLD9)/D
      PBLB = (BLB + R2*BNB)/D
      PBLP = (BLP + R2*BNP)/D
      PBLR = (BLR + R2*BNR)/D
```

```
      PBLD1 = (BLD1 + R2*BND1)/D
      PBLD2 = (BLD2 + R2*BND2)/D
      PBLD3 = (BLD3 + R2*BND3)/D
      PBLD4 = (BLD4 + R2*BND4)/D
      PBLD5 = (BLD5 + R2*BND5)/D
      PBLD6 = (BLD6 + R2*BND6)/D
      PBLD7 = (BLD7 + R2*BND7)/D
      PBLD8 = (BLD8 + R2*BND8)/D
      PBLD9 = (BLD9 + R2*BND9)/D
C
      PBYB = BYB/U
      PBYP = SAL
      PBYR = -CAL
      PBYPHI = 32.2*CTH/U
      PBYD1 = BYD1/U
      PBYD2 = BYD2/U
      PBYD3 = BYD3/U
      PBYD4 = BYD4/U
      PBYD5 = BYD5/U
      PBYD6 = BYD6/U
      PBYD7 = BYD7/U
      PBYD8 = BYD8/U
      PBYD9 = BYD9/U
C
C( LATERAL DIRECTIONAL STATE MATRIX
C
C
      DO 1805 I=1,5
       DO 1806 J=1,5
 1806  DIRMAT(I,J)=0.0
 1805 CONTINUE
      DIRMAT(1,3)=1.0
      DIRMAT(2,1)=PBYPHI
      DIRMAT(2,2)=PBYB
      DIRMAT(2,3)=PBYP
      DIRMAT(2,4)=PBYR
      DIRMAT(2,5)=32.2*STH/U
      DIRMAT(3,2)=PBLB
      DIRMAT(3,3)=PBLP
      DIRMAT(3,4)=PBLR
      DIRMAT(4,2)=PBNB
      DIRMAT(4,3)=PBNP
      DIRMAT(4,4)=PBNR
      DIRMAT(5,4)=1.0
C
C
C      OUTPUT THE STATE MATRIX
C
C
      WRITE(*,830)
      WRITE(*,1810)
 1810 FORMAT('1',2X,'LATERAL DIRECTIONAL STATE MATRIX')
      WRITE(*,1820)
```

```
 1820 FORMAT('0',5X,'STATES = PHI,BETA,P,R,PSI')
      WRITE(*,*)
      WRITE(*,1825) (DIRMAT(1,I),I=1,5)
      WRITE(*,1825) (DIRMAT(2,I),I=1,5)
      WRITE(*,1825) (DIRMAT(3,I),I=1,5)
      WRITE(*,1825) (DIRMAT(4,I),I=1,5)
      WRITE(*,1825) (DIRMAT(5,I),I=1,5)
      WRITE(*,*)
 1825 FORMAT('0',2X,5(G11.4,4X) )
C
C      LATERAL DIRECTIONAL INPUT MATRIX
C
C
      DO 1830 I=1,9
      DIRBMAT(1,I)=0.0
      DIRBMAT(5,I)=0.0
 1830 CONTINUE
      DIRBMAT(2,1)=PBYD1
      DIRBMAT(2,2)=PBYD2
      DIRBMAT(2,3)=PBYD3
      DIRBMAT(2,4)=PBYD4
      DIRBMAT(2,5)=PBYD5
      DIRBMAT(2,6)=PBYD6
      DIRBMAT(2,7)=PBYD7
      DIRBMAT(2,8)=PBYD8
      DIRBMAT(2,9)=PBYD9
      DIRBMAT(3,1)=PBND1
      DIRBMAT(3,2)=PBND2
      DIRBMAT(3,3)=PBND3
      DIRBMAT(3,4)=PBND4
      DIRBMAT(3,5)=PBND5
      DIRBMAT(3,6)=PBND6
      DIRBMAT(3,7)=PBND7
      DIRBMAT(3,8)=PBND8
      DIRBMAT(3,9)=PBND9
      DIRBMAT(4,1)=PBLD1
      DIRBMAT(4,2)=PBLD2
      DIRBMAT(4,3)=PBLD3
      DIRBMAT(4,4)=PBLD4
      DIRBMAT(4,5)=PBLD5
      DIRBMAT(4,6)=PBLD6
      DIRBMAT(4,7)=PBLD7
      DIRBMAT(4,8)=PBLD8
      DIRBMAT(4,9)=PBLD9
C
C     PRINT OUT THE INPUT MATRIX
C
      WRITE(*,1850)
 1850 FORMAT('0',2X,'LATERAL DIRECTIONAL INPUT MATRIX')
      WRITE(*,1860)
 1860 FORMAT('0',4X,'FOR INPUTS: DEL1=RUDDER,DEL2=DIFF CAN')
      WRITE(*,1870)
 1870 FORMAT( 6X,'DEL3=DIFF STAB, DEL4=DIFF AIL, DEL5=DIFF TEF')
```

```
      WRITE(*,1880)
1880  FORMAT(6X,'DEL6 TO 9 ARE REVERSER VANE PORTS')
      WRITE(*,1890)
1890  FORMAT('0',5X,'ROW1',11X,'ROW2',11X,'ROW3',11X,'ROW4',11X,'ROW5')
      DO 1900 I=1,9
      WRITE(*,1825) (DIRBMAT(J,I),J=1,5)
1900  CONTINUE
      GO TO 421
450   CONTINUE
      END
```

B-3 Sample Program Execution

```
/gt,stolbin
FILE STOLBIN RETRIEVED
/ ***************************************************
 *** STABILITY DERIVATIVE TRANSFORMATION PROGRAM  ***
 ***************************************************
 ENTER BODY AXIS (NON-DIMENSIONALIZED) COEFFICIENTS
 FOR TRANSFORMATION TO DIMENSIONALIZED BODY AXIS
 AND TO GENERATE STATE AND INPUT MATRICES.
 NOTE:  ALL COEFFICIENTS ARE REQUESTED WHEN COMPUTING
 ***************************************************
 TO TRANSFORM ONLY LONGITUDINAL DATA - TYPE LONG
 TO TRANSFORM ONLY LATERAL-DIRECTIONAL DATA - TYPE LAT
 TO TRANSFORM BOTH LONG AND LAT-DIR DATA - TYPE BOTH
 KEYWORD = long
  ***************************************************
?  Q  (DYNAMIC PRESSURE - LBS/FT**2) = 48.06
?  S  (WING REFERENCE AREA - FT**2) = 608
?  C  (WING MEAN AFRODYNAMIC CORD - FT) = 15.9399
?  B  (WING SPAN - FT) = 42.7
?  VT (TRIM VELOCITY - FT/SEC) = 200
?  THETA (PITCH ANGLE - DEGS) = 11.5840
?  W  (WEIGHT - LBS) = 33576
?  INERTIAS MUST BE INPUT IN BODY AXIS.
?  IXX  (SLUG-FT**2) =  23634
?  IYY  (SLUG-FT**2) = 181837
?  IZZ  (SLUG-FT**2) = 199674
?  IXZ  (SLUG-FT**2) = -3086
?  ***************************************************
                AIRCRAFT PARAMETERS
 Q  (DYNAMIC PRESSURE - LBS/FT**2) =   48.0600
 S  (WING REFERENCE AREA - FT**2) =   608.000
 C  (WING MEAN AERODYNAMIC CORD - FT) =   15.9399
 B  (WING SPAN - FT) =   42.7000
 VT (TRIM VELOCITY - FT/SEC) =   200.000
 THETA =   11.5840
 W  (WEIGHT - LBS) =   33576.0
 IXX  (SLUG-FT**2) =   23634.0
 IYY  (SLUG-FT**2) =   181837.
 IZZ  (SLUG-FT**2) =   199674.
 IXZ  (SLUG-FT**2) =  -3086.00
 ***************************************************
 IS THE ENTERED DATA CORRECT ?  (YES/NO)
?  yes
***************************************************
?  ALPHA (DEG) = 11.5840
?  CZA = -.0796233
?  CXA = .208763
?  CMA = .931356e-02
?  CZQ = 0
?  CXQ = 0
?  CMQ = -.169490
?  CZU = .658418e-2
?  CXU = -.246580
```

```
?  CMU = -.0140683
?  CZH = -.181866e-4
?  CXH = .681123e-3
?  CMH = -.388591e-4
?  CZD1 = -.257164e-2
?  CXD1 = -.153014e-2
?  CMD1 = .57288e-2
?  CZD2 = -.955232e-2
?  CXD2 = -.201656e-2
?  CMD2 = -.107546e-1
?  CZD3 = -.4888427e-2
?  CXD3 = .136541e-2
?  CMD3 = .112899e-2
?  CZD4 = -.451559e-2
?  CXD4 = .925632e-3
?  CMD4 = -.214211e-2
?  CZD5 = .135028e-2
?  CXD5 = -.340353e-2
?  CMD5 = .129075e-2
?  CZD6 = -.135028e-2
?  CXD6 = -.340353e-2
?  CMD6 = -.137616e-2
?  CZD7 = .135028e-2
?  CXD7 = -.340353e-2
?  CMD7 = .129075e-2
?  CZD8 = -.135028e-2
?  CXD8 = -.340353e-2
?  CMD8 = -.137616e-2
  ******************************************************
                ALPHA =  11.5840
      LONGITUDINAL NON-DIM BODY AXIS COEFFICIENTS(1/DEG)
CZA = -.796233E-01     CMA = .931356E-02    CXA = .208763E-03
CZQ =  0.              CMQ =  -.169490      CXQ =  0.
CZH = -.181866E-04     CMH = -.388591E-04   CXH = .681123E-03
CZU = .658418E-02      CMU = -.140683E-01   CXU =  -.246580
CZD1= -.257164E-02    CMD1 = .572880E-02   CXD1 =-.153014E-02
CZD2= -.955232E-02    CMD2 = -.107546E-01  CXD2 =-.201656E-02
CZD3= -.488427E-02    CMD3 =  .112899E-02  CXD3 = .136541E-02
CZD4= -.451559E-02    CMD4 = -.214211E-02  CXD4 = .925632E-03
CZD5= .135028E-02     CMD5 = .129075E-02   CXD5 =-.340353E-02
CZD6=-.135028E-02     CMD6 = -.137616E-02  CXD6 =-.340353E-02
CZD7= .135028E-02     CMD7 =  .129075E-02  CXD7 =-.340353E-02
CZD8=-.135028E-02     CMD8 = -.137616E-02  CXD8 =-.340353E-02
 ******************************************************
 IS THE ENTERED DATA CORRECT ?   (YES/NO)
? yes
******************************************************
     LONGITUDINAL AXIS DIMENSIONAL DERIVATIVES
                BODY AXIS (1/RAD)
ZA =  -127.843         MA =   1.36688       XA =   .335190
ZQ =  0.               MQ =  -.991250       XQ =  0.
ZH =  -.254821E-05     MH =  -.497684E-06   XH =   .954355E-04
ZU =   .184508E-02     MU =  -.360356E-03   XU =  -.690991E-01
```

```
ZD1=  -4.12902        MD1 =   .840770      XD1 =  -2.45679
ZD2=  -15.3372        MD2 = -1.57837       XD2 =  -3.23779
ZD3=  -7.84218        MD3 =  .165693       XD3 =   2.19230
ZD4=  -7.25022        MD4 = -.314380       XD4 =   1.48619
ZD5=   2.16801        MD5 =  .189433       XD5 =  -5.46470
ZD6=  -2.16801        MD6 = -.201968       XD6 =  -5.46470
ZD7=   2.16801        MD7 =  .189433       XD7 =  -5.46470
ZD8=  -2.16801        MD8 = -.201968       XD8 =  -5.46470
*****************************************************

1     LONGITUDNAL STATE MATRIX(BODY AXIS)

0  FOR STATE1=U,STATE2=Q,STATE3=ALPHA,STATE4=THETA

0 -.690991E-01    -40.1609         .335190      -31.5441
0 -.360356E-03    -.991250        1.36688        0.
0  .922542E-05     .979631       -.639215      -.323295E-01
0 0.               1.00000       0.              0.


0     LONGITUDNAL INPUT MATRIX

  FOR DEL1=CANARD,DEL2=STAB,DEL3=TEF,DEL4=DR AILERON
      DEL5=RT RV, DEL6=RB RV, DEL7=LT RV, DEL8=LB RV


0     ROW1            ROW2            ROW3            ROW4

   -2.45679         .840770       -.206451E-01    0.
   -3.23779       -1.57837        -.766859E-01    0.
    2.19230         .165693       -.392109E-01    0.
    1.48619       -.314380        -.362511E-01    0.
   -5.46470         .189433        .108400E-01    0.
   -5.46470       -.201968        -.108400E-01    0.
   -5.46470         .189433        .108400E-01    0.
   -5.46470       -.201968        -.108400E-01    0.

*****************************************************
IS ANOTHER PROGRAM RUN DESIRED ?   (YES/NO)
```

APPENDIX C. STOL F-15 DATA

This appendix presents the data used in this study. The data shown has been reduced via the program STOLCAT (Appendix B) and is given only for the longitudinal mode. The original data from McDonnell Douglas [23] consisted of a list of aircraft size, weight, structural parameters, and nondimensional stability derivatives for linearized equations of motion. Data were given for eight flight conditions, four of which are used in this study. The data for the first flight condition, about which the controller is designed, is listed in Appendix B. The remaining data points; for the aircraft at sea level, normal landing gross weight, at 100 knots; sea level, heavy landing gross weight, at 120 knots; and at 10,000 feet altitude, normal gross weight, at 120 knots; are listed respectively in this appendix.

```
**************************************************
                AIRCRAFT PARAMETERS
Q  (DYNAMIC PRESSURE - LBS/FT**2) =    39.6400
S  (WING REFERENCE AREA - FT**2) =    608.000
C  (WING MEAN AERODYNAMIC CORD - FT) =    15.9400
B  (WING SPAN - FT) =    42.7000
VT (TRIM VELOCITY - FT/SEC) =    168.000
THETA =    17.3952
W  (WEIGHT - LBS) =    33576.0
IXX  (SLUG-FT**2) =    23644.0
IYY  (SLUG-FT**2) =    181847.
IZZ  (SLUG-FT**2) =    199674.
IXZ  (SLUG-FT**2) =   -3086.00
**************************************************
               ALPHA =  17.3952
     LONGITUDINAL NON-DIM BODY AXIS COEFFICIENTS(1/DEG)
CZA = -.666000E-01    CMA = .845000E-02   CXA = .168000E-01
CZQ = 0.              CMQ =-.177000       CXQ =  0.
CZH = -.211000E-03    CMH = .200000E-03   CXH = .103000E-02
CZU = -1.47000        CMU = .864000E-01   CXU = .438000E-01
CZD1= -.408000E-02    CMD1=  .615000E-02  CXD1=-.131000E-02
CZD2= -.114000E-01    CMD2= -.114000E-01  CXD2= .159700E-03
CZD3= -.561000E-02    CMD3=  .431000E-03  CXD3= .176000E-02
CZD4= -.409000E-02    CMD4= -.142000E-02  CXD4= .128000E-02
CZD5=  .290000E-02    CMD5=  .284000E-02  CXD5=-.420000E-02
CZD6= -.157000E-02    CMD6= -.242000E-02  CXD6=-.488000E-02
CZD7=  .290000E-02    CMD7=  .384000E-02  CXD7=-.420000E-02
CZD8= -.157000E-02    CMD8= -.242000E-02  CXD8=-.488000E-02
**************************************************
1     LONGITUDNAL STATE MATRIX(BODY AXIS)

  FOR STATE1=U,STATE2=Q,STATE3=ALPHA,STATE4=THETA

.120520E-01     -50.2254          22.2483       -30.7273
.217297E-02     -1.01640          1.02282         0.
-.240765E-02    .954265         -.524991        -.573008E-01
   0.              1.00000        0.              0.

0     LONGITUDNAL INPUT MATRIX

  FOR DEL1=CANARD,DEL2=STAB,DEL3=TEF,DEL4=DR AILERON
      DEL5=RT RV, DEL6=RB RV, DEL7=LT RV, DEL8=LB RV

    ROW1            ROW2             ROW3            ROW4

   -1.73483         .744418        -.321616E-01     0.
    .211491        -1.37990        -.898633E-01     0.
    2.33077         .521698E-01    -.442222E-01     0.
    1.69511        -.171882        -.322404E-01     0.
   -5.56207         .343764         .228600E-01     0.
   -6.46259        -.292926        -.123759E-01     0.
   -5.56207         .464808         .228600E-01     0.
   -6.46259        -.292926        -.123759E-01     0.
```

```
**************************************************
                AIRCRAFT PARAMETERS
 Q  (DYNAMIC PRESSURE - LBS/FT**2) =   48.8700
 S  (WING REFERENCE AREA - FT**2) =   608.000
 C  (WING MEAN AERODYNAMIC CORD - FT) =   15.9390
 B  (WING SPAN - FT) =   42.7000
 VT (TRIM VELOCITY - FT/SEC) =   201.100
 THETA =   15.4400
 W  (WEIGHT - LBS) =   43511.0
 IXX  (SLUG-FT**2) =   35215.0
 IYY  (SLUG-FT**2) =   190800.
 IZZ  (SLUG-FT**2) =   219105.
 IXZ  (SLUG-FT**2) =  -2881.00
 **************************************************
               ALPHA =  15.4400
      LONGITUDINAL NON-DIM BODY AXIS COEFFICIENTS(1/DEG)
CZA = -.784900E-01   CMA =  .957400E-02   CXA = .150900E-02
CZQ =  0.            CMQ = -.169000       CXQ =  0.
CZH = -.167600E-03   CMH =  .176600E-03   CXH = .666200E-03
CZU = -1.06500       CMU =  .639400E-01   CXU =-.619300E-02
CZD1= -.263600E-02   CMD1=  .557600E-02   CXD1=-.155200E-02
CZD2= -.831500E-02   CMD2= -.102000E-01   CXD2=-.274900E-03
CZD3= -.559100E-02   CMD3=  .852100E-03   CXD3= .115700E-02
CZD4= -.450800E-02   CMD4= -.211100E-02   CXD4= .942100E-03
CZD5=  .189600E-02   CMD5=  .255400E-02   CXD5=-.312000E-02
CZD6= -.742200E-03   CMD6= -.130100E-02   CXD6=-.359500E-02
CZD7=  .189600E-02   CMD7=  .255400E-02   CXD7=-.312000E-02
CZD8= -.742200E-03   CMD8= -.130100E-02   CXD8=-.359500E-02
 **************************************************
1     LONGITUDNAL STATE MATRIX(BODY AXIS)

0  FOR STATE1=U,STATE2=Q,STATE3=ALPHA,STATE4=THETA

 -.135432E-02    -53.5387        1.90114       -31.0379
  .157841E-02    -.952482        1.36158         0.
 -.115813E-02     .963910       -.491731       -.426284E-01
   0.              1.00000        0.              0.

0     LONGITUDNAL INPUT MATRIX

  FOR DEL1=CANARD,DEL2=STAB,DEL3=TEF,DEL4=DR AILERON
      DEL5=RT RV, DEL6=RB RV, DEL7=LT RV, DEL8=LB RV

0     ROW1            ROW2            ROW3            ROW4

   -1.95532         .793002       -.165142E-01    0.
   -.346338        -1.45061       -.520925E-01    0.
    1.45767         .121183       -.350270E-01    0.
    1.18692        -.300220       -.282421E-01    0.
   -3.93079         .363222        .118782E-01    0.
   -4.52923        -.185024       -.464980E-02    0.
   -3.93079         .363222        .118782E-01    0.
   -4.52923        -.185024       -.464980E-02    0.
```

```
**************************************************
              AIRCRAFT PARAMETERS
Q  (DYNAMIC PRESSURE - LBS/FT**2) =   35.1200
S  (WING REFERENCE AREA - FT**2) =   608.000
C  (WING MEAN AERODYNAMIC CORD - FT) =   15.9400
B  (WING SPAN - FT) =   42.7000
VT (TRIM VELOCITY - FT/SEC) =   200.000
THETA =   16.4279
W  (WEIGHT - LBS) =   33576.0
IXX  (SLUG-FT**2) =   23644.0
IYY  (SLUG-FT**2) =   181847.
IZZ  (SLUG-FT**2) =   199674.
IXZ  (SLUG-FT**2) =  -3086.00
**************************************************
              ALPHA =  16.4279
     LONGITUDINAL NON-DIM BODY AXIS COEFFICIENTS(1/DEG)
CZA = -.687596E-01   CMA =  .706499E-02   CXA = .179080E-01
CZQ =  0.            CMQ = -.176840       CXQ =  0.
CZH = -.212076E-03   CMH =  .184559E-03   CXH = .116490E-02
CZU = -1.43120       CMU =  .623987E-01   CXU = .491830E-01
CZD1= -.417763E-02   CMD1=  .606027E-02   CXD1=-.126310E-02
CZD2= -.115909E-01   CMD2= -.114406E-01   CXD2=-.395643E-03
CZD3= -.553756E-02   CMD3=  .675633E-03   CXD3= .172800E-02
CZD4= -.422113E-02   CMD4= -.157480E-02   CXD4= .124458E-02
CZD5=  .245366E-02   CMD5=  .319344E-02   CXD5=  0.
CZD6= -.161307E-02   CMD6= -.231230E-02   CXD6=-.354056E-02
CZD7=  .245366E-02   CMD7=  .319344E-02   CXD7=-.300115E-02
CZD8= -.161307E-02   CMD8= -.231298E-02   CXD8=-.354056E-02
**************************************************
1     LONGITUDNAL STATE MATRIX(BODY AXIS)

0  FOR STATE1=U,STATE2=Q,STATE3=ALPHA,STATE4=THETA

 .100716E-01     -56.5617        21.0114        -30.8855
 .116793E-02     -.755739         .757660           0.
-.146540E-02      .959176        -.403377        -.455322E-01
    0.            1.00000           0.             0.

0     LONGITUDNAL INPUT MATRIX

  FOR DEL1=CANARD,DEL2=STAB,DEL3=TEF,DEL4=DR AILERON
      DEL5=RT RV, DEL6=RB RV, DEL7=LT RV, DEL8=LB RV

    ROW1            ROW2            ROW3            ROW4

  -1.48199         .649912       -.245080E-01      0.
  -.464206        -1.22691       -.679978E-01      0.
   2.02746         .724559E-01   -.324860E-01      0.
   1.46026        -.168884       -.247632E-01      0.
   0.              .342469        .143943E-01      0.
  -4.15413        -.247974       -.946304E-02      0.
  -3.52124         .342469        .143943E-01      0.
  -4.15413        -.248047       -.946304E-02      0.
```

APPENDIX D. FOUR-INPUT/FOUR-OUPUT MODEL

D-1 Introduction

The controller designed in the main body of this study is for a three-input/three-output system. This has been done so the results would be directly comparible to the results of similar design efforts done via the Porter method [1]. As this is not a limitation of the design procedure, an example of a higher dimension system is given in this appendix. It should be noted that this controller was designed in one CGTPIF session and that it was not possible to evaluate the controller with ODEF15, as that software could not be easily modified for the increase in dimensions. Therefore, only a time history of the output variables and the control inputs (from CGTPIF) are presented.

D-2 The Design Set Up

For the purposes of this design, it was decided to use the flaps and aileron, decoupled from the canard, as the additional input, and angle of attack as the additional output. As a result the A matrix of Equation (4-3) is unchanged, but the control input matrix becomes:

$$B = \begin{bmatrix} -2.45679 & -3.23779 & 2.58234 & -21.8588 \\ -.84077 & -1.57837 & -.231534 & -.02507 \\ -.020645 & -.0766859 & -.055866 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (D-1)$$

and the associated output matrix is:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \qquad \text{(D-2)}$$

All other design model matrices remain as in Chapter 4.

The implicit model becomes

$$A_m = \begin{bmatrix} -.05 & 0 & 0 & 0 \\ 0 & -.05 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -.5 \end{bmatrix} \qquad \text{(D-3)}$$

with the B_m, C_m, and D_m matrices still zero. The A_m matrix is the same as seen in Equation (5-9) with angle of attack inserted as the third state and flight path angle now the fourth state. The implicit weightings were extrapolated from the three-input/ three-output model and, after a very few iterations, were finalized at 1:1:1:4 for the output derivatives of the velocity, pitch rate, angle of attack, and flight path angle, respectively; and 2:2:2:1 for the pseudocontrol rates for the canard, stabilator, flap/ aileron combination, and reverser vanes, in that order. The only change to the explicit model is that the C_m matrix is now the four by four matrix:

$$C_m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{(D-4)}$$

which has the same characteristics as C_m of Equation (4-13).

The truth model changes in that the actuator dynamics of the flap/aileron combination are now added to the A_t matrix (Equation 4-16), increasing that to an eleven by eleven matrix, with a proportionate increase in the dimensions of the other truth model matrices.

D-3 Final Controller for the Four-Input/Four-output System

After a very few iterations through CGTPIF, the following gain matrices were developed for the four-input/four-output system.

$$K_x = \begin{bmatrix} .01184 & -10.41 & 95.65 & -.2047 \\ -.003348 & -9.706 & -2.068 & -.1432 \\ -.006805 & 22.27 & -396.6 & .5563 \\ -3.401 & 5.446 & -57.84 & .1119 \end{bmatrix} \quad (D-5)$$

$$K_z = \begin{bmatrix} .0018015 & -2.69 & 1.581 & -.2615 \\ .008313 & -.3615 & -1.042 & -1.283 \\ .001909 & 9.342 & -7.304 & -.3133 \\ -.08332 & 1.579 & -.9434 & .1275 \end{bmatrix} \quad (D-6)$$

$$K_{xm} = \begin{bmatrix} .011649 & -17.2 & .04566 & -.01594 \\ -.003493 & -9.778 & -.1395 & -.009605 \\ -.006351 & 43.09 & -.1513 & .036301 \\ -3.404 & 6.826 & -1.443 & .006357 \end{bmatrix} \quad (D-7)$$

$$K_{xu} = \begin{bmatrix} -.0093608 \\ -.0063333 \\ .018368 \\ .0037175 \end{bmatrix} \qquad \text{(D-8)}$$

The time-response for this controller can be seen in Figures D-1 and D-2. This response is similar to those for the three-input/three-output system (Figs. 5-7 and 5-8), but the overshoot is slightly smaller. The advantage to having control over an additional input and output is that one may exert tighter control over the system and also independent control over more separate channels (variables), so that compromises in overall performance (and robustness) ought to be less severe. One must also take into account the additional computation time involved with the higher dimensionality and the trade-offs between accuracy and computation time must be considered.

FINAL 4-INPUT/4-OUTPUT CONTROLLER

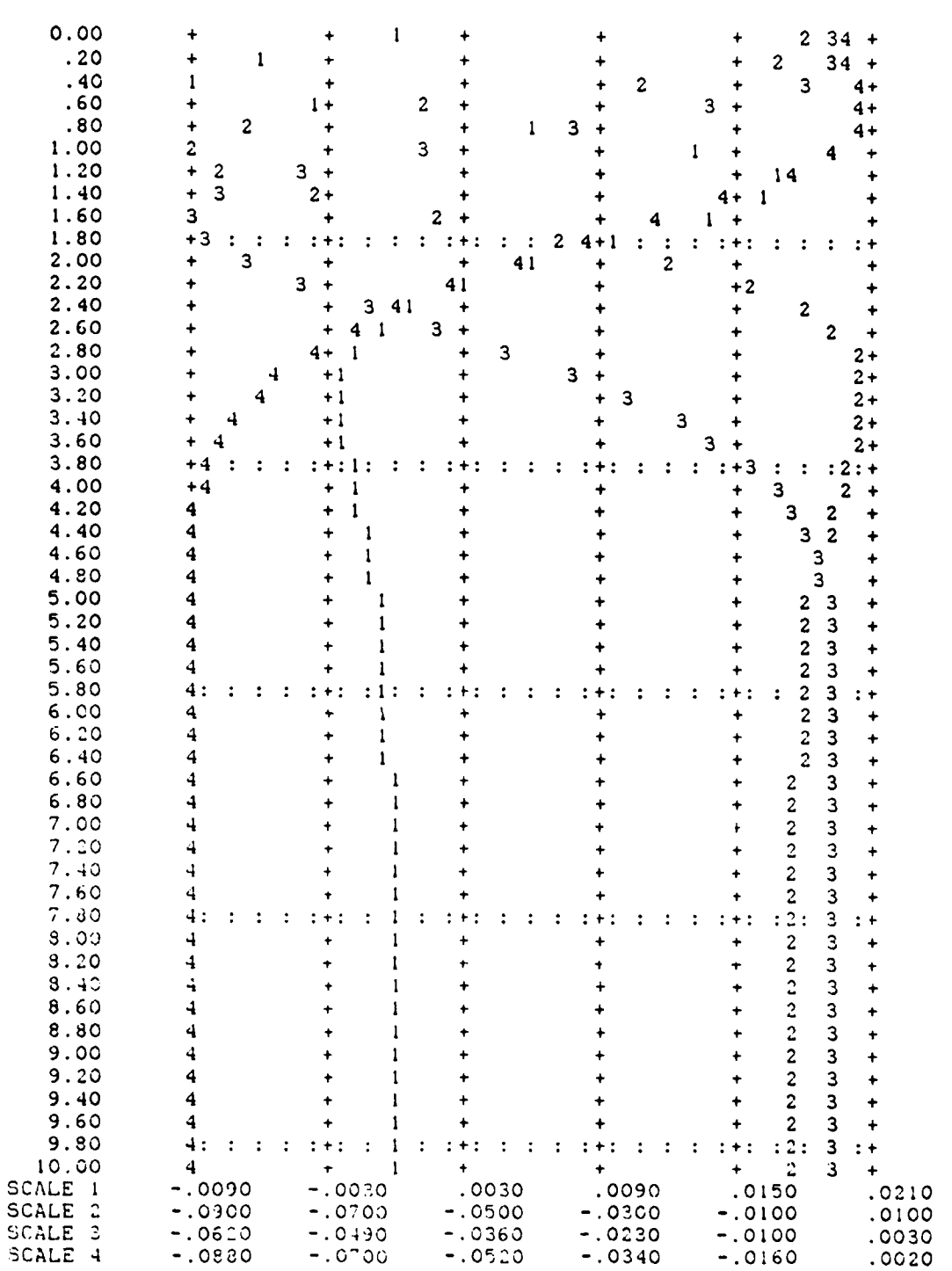


Figure D-1. Output Variables for a Four-Input/ Four-Output System (1) Velocity (fps) (2) Pitch Rate (rad/sec) (3) Flight Path Angle (rad)

FINAL 4-INPUT/4-OUTPUT CONTROLLER

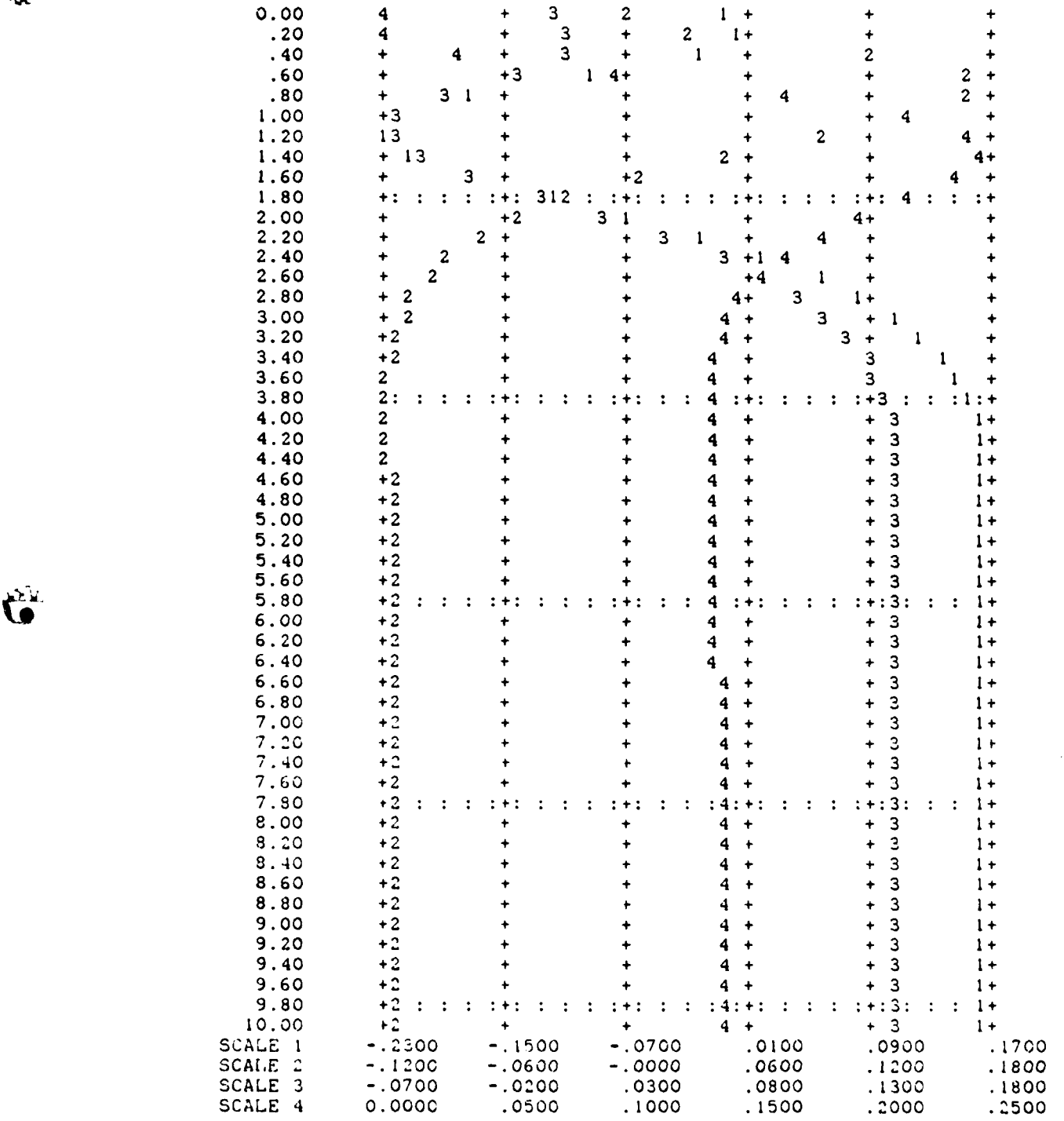


Figure D-2. Control Deflections for a Four-Input/ Four-Output System (1) Canard (2) Stabilator (3) Reverser Vanes (radians)

APPENDIX E. ACTUATOR COMBINATION AND ORDER REDUCTION

E-1 Introduction

This appendix considers the actuators of the STOL F-15. There are two main concerns with the actuators in this study. First, as there are more system inputs than outputs, and as CGTPIF requies these to be equal, a number of inputs must either be combined, excluded from consideration, or the number of outputs to be controlled could be increased to match the number of inputs available. This last alternative was not considered for the majority of this study to keep the results comparable with those achieved using other design methods [1,7]. For this study it was decided to combine the canard with the flaps and the aileron rather than delete the inputs from the latter two. Secondly, the dynamics of the actuators must be taken into account in any evaluation, if not the design, of a control law. An examination of possible reduction of the order of these dynamics is also considered in this appendix.

E-2 Combination of Actuators

Each control surface contributes a given amount to the generation of moments about the aircraft center of gravity. If the contribution of a surface is neglected, the controller designed, or evaluated, is suboptimal. For the purpose of this study, rather than delete the inputs for the flaps and ailerons, the effects of these control surfaces will be combined with the canard. To accomplish

this combination, two factors are considered, position and rate limits, in scaling the contribution of each surface to the combined input. In this case the rate limit for the canard, 23 degrees per second, is by far the more restrictive limit. For position limits, it is desired that an input that would saturate a control surface would saturate all of the combined surfaces simultaneously. To accomplish this, the contribution of each surface is weighted as to the minimum distance it can travel before saturating. For the canard/flap/aileron combination of the STOL F-15, the ratio for these inputs is 1:.16667:.5. The terms of the control input matrix (see data listing in Appendix A) for each of the appropriate control surfaces is multiplied by the given weight, the products are then added together, and the resulting sum is then inserted into the first row of the B matrix. The resulting control input matrix is shown in Equation (4-4).

E-3 Order Reduction of Actuator Dynamics

The original actuator models for the stabilator and canard, as provided by McDonnell Douglas [23], are third order with a real pole at -30.62 and a complex pair at $-138.63 \pm j\ 235.06$. To reduce the order of the actuator model to second order, frequency domain analysis is performed on the third order model and a proposed second order replacement is generated. Figure E-1 demonstrates that the reduced order model has the same low frequency

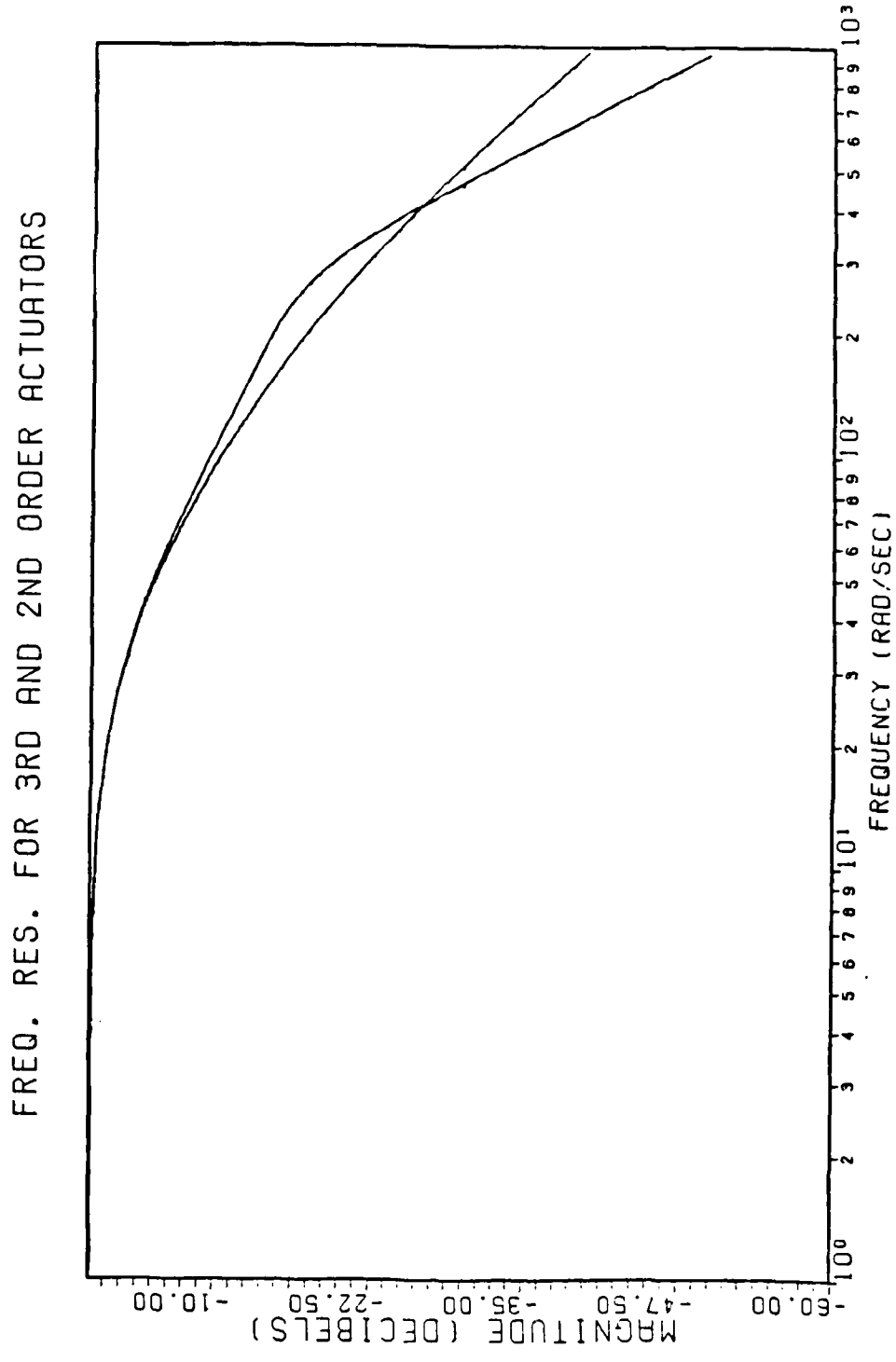


Figure E-1. Frequency Response for Second and Third Order Actuators

characteristics as the third order model and that it is not until the high frequency band that the models show widely differing responses and these high frequency effects can be neglected as being outside the bandwidth of interest. Therefore, the second order model is assumed to be a proper substitute for the stabilator and the canard actuators. This reduced order model is used to ease computational loading for designs that include actuator dynamics in the design model, which is not the case in this study, or to reduce the order of the truth model used for controller evaluation, once again for computation efficiency.

VITA

Gregory Louis Gross was born on 2 May 1955 in Canton, Ohio and graduated from Bay High School, Bay Village, Ohio in 1973. In 1977 he received a Bachelor of Science in Aeronautical Engineering from the U. S. Air Force Academy and was commissioned in the Air Force. Prior to entering pilot training, he was temporarily assigned to the Air Force Test and Evaluation Center where he worked on test plans and test reports for the F-4G Wild Weasel and the EF-111 Tactical Jamming System. He completed pilot training at Williams AFB, Arizona in November 1978. He was then assigned to the 43rd Air Refueling Squadron at Fairchild AFB, Washington first as a copilot and then as an aircraft commander in the KC-135. In May 1983 he received the degree of Master of Business Administration from Gonzaga University in Spokane, Washington. In May 1984, he entered the School of Engineering, Air Force Institute of Technology.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GAE/ENG/85D-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering Air Force Institute of Tech 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Flight Dynamics Lab 8b. OFFICE SYMBOL (If applicable) AFWAL/FIGX	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) LQG/LTR Design of a Robust Flight Controller for the Stol F-15	
12. PERSONAL AUTHOR(S) Gregory L. Gross Captain, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 85 December 15. PAGE COUNT
16. SUPPLEMENTARY NOTATION	Approved for public release: IAW AFR 190-1. LYNN E. WOLAVER 16 JAN 86 Dean for Research and Professional Development Air Force Institute of Technology (AFIT) Wright-Patterson AFB OH 45433
17. COSATI CODES FIELD: 1 / GROUP: 3 / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) LQG/LTR, Kalman Filter, Multivariable Control, Model Following Controller, STOL,

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A robust controller for the approach and landing phase of the Short Take-off and Landing (STOL) F-15 is developed via LQG/LTR (Linear System model, Quadratic cost, Gaussian models of uncertainty, used for controller synthesis, with Loop Transmission Recovery techniques of tuning the filter in the loop for control robustness enhancement) methods. Reduced-order full-state feedback controllers are synthesized using CGT/PI (Command Generator Tracking feedforward conpensator to incorporate handling qualities, with Proportional plus Integral feedback) synthesis, specifically using implicit model following to provide good robustness characteristics in the full-state feedback case. The robustness is fully assessed using realistic simulations of the real-world system with meaningful deviations from design conditions. Once a Kalman filter is embedded into the loop to estimate states rather than assuming artificial access to all states, LTR methodology is used to preserve as much robustness as possible. A full assessment of performance and robustness of these final implementable designs is provided.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Peter S. Maybeck, BS, PHD	22b. TELEPHONE NUMBER (Include Area Code) (513) 255-3576 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

END

FILMED

3 -86

DTIC